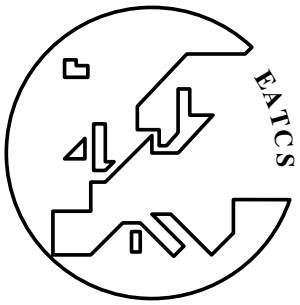
44th International Colloquium on Automata, Languages, and Programming

ICALP 2017, Warsaw, Poland, July 10–14, 2017

Edited by
Ioannis Chatzigiannakis
Piotr Indyk
Fabian Kuhn
Anca Muscholl



Editors

Ioannis Chatzigiannakis
Department of Computer, Control, and Management Engineering
Sapienza University of Rome
Italy
ichatz@dis.uniroma1.it

Piotr Indyk
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
USA
indyk@mit.edu

Fabian Kuhn
Institut für Informatik
Albert-Ludwigs-Universität
Germany
kuhn@cs.uni-freiburg.de

Anca Muscholl
LaBRI
Université Bordeaux
France
anca@labri.fr

ACM Classification 1998
F. Theory of Computation

ISBN 978-3-95977-041-5

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-041-5.

Publication date
July, 2017

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

Digital Object Identifier: 10.4230/LIPIcs.ICALP.2017.0

ISBN 978-3-95977-041-5 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

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

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

This volume contains the papers presented at ICALP 2017, the 44th edition of the International Colloquium on Automata, Languages and Programming, held in Warsaw, Poland during July 10–14, 2017. ICALP is a series of annual conferences of the European Association for Theoretical Computer Science (EATCS), which first took place in 1972. This year, the ICALP program consisted of three tracks:

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

In response to the call for papers, a total 459 submissions were received: 296 for track A, 108 for track B, and 55 for track C. Each submission was reviewed by at least three Program Committee members, aided by many subreviewers. Out of these, the committee decided to accept 137 papers for inclusion in the scientific program: 88 papers for Track A, 32 for Track B, and 17 for Track C. The selection was made by the Program Committees based on originality, quality, and relevance to theoretical computer science. The quality of the manuscripts was very high, and many deserving papers could not be selected.

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

The best paper awards were given to the following papers:

- Track A: Andreas Björklund, Petteri Kaski and Ioannis Koutis. "Directed Hamiltonicity and Out-Branchings via Generalized Laplacians".
- Track B: Michael Benedikt, Pierre Bourhis and Michael Vanden Boom. "Characterizing Definability in Decidable Fixpoint Logics".
- Track C: Eyjólfur Ingi Ásgeirsson, Magnus M. Halldorsson and Tigran Tonoyan. "Universal Framework for Wireless Scheduling Problems".

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

- Track A: Euiwoong Lee. "Improved Hardness for Cut, Interdiction, and Firefighter Problems".
- Track B: Fabian Reiter. "Asynchronous Distributed Automata: A Characterization of the Modal Mu-Fragment".

Apart from the contributed talks, ICALP 2017 included invited presentations by Mikołaj Bojańczyk, Monika Henzinger, Ronitt Rubinfeld and Mikkel Thorup. This volume of the proceedings contains all contributed papers presented at the conference together with the papers and abstracts of the invited speakers.

The program of ICALP 2017 also included presentation of the EATCS Award 2017 to Eva Tardos, the Presburger Award 2017 to Alexandra Silva and the EATCS Distinguished Dissertation Award to Vincent Cohen-Addad, Mika Göös and Steen Vester.

Three satellite events of ICALP were held on 14 July, 2017:

- SSG: Algorithms and Structure for Sparse Graphs
- AVeRTS: Algorithmic Verification of Real-Time Systems
- SP: Separability Problems

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The Lipa Summer School was organized on topics connected to logic in computer science during 3–6 July, 2017.

We wish to thank all authors who submitted extended abstracts for consideration, the Program Committees for their scholarly effort, and all referees who assisted the Program Committees in the evaluation process. We are also grateful to Mikołaj Bojańczyk, Piotr Sankowski, Bartek Klin and Filip Murlak for organizing ICALP 2017 and all the support staff of the Organizing Committee, especially Hanna Bargieł and Alicja Kosińska from Global Congress.

We would like to thank Paul Spirakis, the president of EATCS, for his generous advice on the organization of the conference.

July 2017

Ioannis Chatzigiannakis
Piotr Indyk
Fabian Kuhn
Anca Muscholl

Organization

Program Committee

Track A

Indyk Piotr	MIT, USA, Chair
Afshani Peyman	Aarhus University, Denmark
Agarwal Pankaj	Duke University, USA
Bringmann Karl	Max Planck Institute for Informatics, Germany
Chattopadhyay Arkadev	Tata Institute of Fundamental Research, India
Chechik Shiri	Tel-Aviv University, Israel
Ene Alina	Boston University, USA
Filmus Yuval	Technion – Israel Institute of Technology, Israel
Gopalan Parikshit	VMware Research, USA
Grossi Roberto	Universita' di Pisa, Italy
Gupta Anupam	Carnegie Mellon University, USA
Ishai Yuval	Technion, Israel
Kapralov Michael	EPFL, Switzerland
Kleinberg Robert	Cornell University, USA
Lu Pinyan	Shanghai University of Finance and Economics, China
Magniez Frederic	CNRS, Univ. Paris Diderot, France
Mahdian Mohammad	Google, USA
Marx Daniel	Computer and Automation Research Institute, Hungarian Academy of Sciences, Hungary
Nanongkai Danupon	KTH Royal Institute of Technology, Sweden, Sweden
Nelson Jelani	Harvard, USA
Pilipczuk Marcin	Institute of Informatics, University of Warsaw, Poland
Sankowski Piotr	University of Warsaw, Poland
Sauerwald Thomas	University of Cambridge, United Kingdom
Scheideler Christian	University of Paderborn, Germany
Sohler Christian	TU Dortmund, Germany
Telikepalli Kavitha	Tata Institute of Fundamental Research, Mumbai, India
Vaikuntanathan Vinod	University of Toronto, Canada
Vegh Laszlo A.	London School of Economics, United Kingdom
Venkatasubramanian Suresh	University of Utah, USA
Vidick Thomas	Caltech, USA
Wee Hoeteck	ENS, France
Weimann Oren	University of Haifa, Israel
Weinberg Seth	MIT, USA

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Track B

Muscholl Anca	LaBRI, Universite Bordeaux, France, Chair
Barcelo Pablo	Universidad de Chile, Chile
Blumensath Achim	Masaryk University, Brno, Czech Republic
Brihaye Thomas	Université de Mons, Belgium
Chatterjee Krishnendu	Institute of Science and Technology (IST), Austria
Coquand Thierry	Chalmers University, Sweden
Dawar Anuj	University of Cambridge, United Kingdom
Endrullis Jörg	Vrije Universiteit Amsterdam, Netherlands
Fisman Dana	University of Pennsylvania, USA
Hofmann Martin	LMU Munich, Germany
Jagadeesan Radha	DePaul University, USA
Kiefer Stefan	University of Oxford, United Kingdom
Kieronski Emanuel	University of Wroclaw, Poland
Kreutzer Stephan	Technical University Berlin, Germany
La Torre Salvatore	Dipartimento di Informatica, Università degli studi di Salerno, Italy
Lin Anthony Widjaja	Department of Computer Science, University of Oxford, United Kingdom
Martens Wim	University of Bayreuth, Germany
Mellies Paul-André	CNRS, Université Paris Diderot, France
Padovani Luca	Università di Torino, Italy
Palamidessi Catuscia	INRIA, France
Pighizzini Giovanni	Dipartimento di Informatica, Università degli Studi di Milano, Italy
Pin Jean-Éric	LIAFA, CNRS and University Paris 7, France
Silva Alexandra	University College London, United Kingdom
Talbot Jean-Marc	LIF, Universite d'Aix-Marseille, France
Viswanathan Mahesh	University of Illinois, Urbana-Champaign, USA
Wilke Thomas	University of Kiel, Germany
Worell James	Oxford University, United Kingdom

Track C

Kuhn Fabian	University of Freiburg, Germany, Chair
Abraham Ittai	VMware Research, USA
Anta Antonio Fernandez	IMDEA Networks Institute, Spain
Aspnes James	Yale, USA
Censor-Hillel Keren	Technion, Israel
Emek Yuval	Technion, Israel
Ghaffari Mohsen	ETH Zurich, Switzerland
Giakkoupis George	INRIA Rennes, France
Gilbert Seth	N/A, Singapore
Haeupler Bernhard	CMU, USA
Korman Amos	CNRS and Université Paris Diderot - Paris 7, France
Kosowski Adrian	IRIF (LIAFA) / Inria Paris, France
Lenzen Christoph	MPI for Informatics, Germany
Masuzawa Toshimitsu	Osaka University, Japan
Panagiotou Konstantinos	University of Munich (LMU), Germany
Panconesi Alessandro	Sapienza University of Rome, Italy
Parter Merav	MIT, CSAIL, USA
Patt-Shamir Boaz	Tel Aviv University, Israel
Pigonlet Yvonne-Anne	ABB Corporate Research, Switzerland
Rajsbaum Sergio	Instituto de Matematicas, UNAM, Mexico
Richa Andrea	Arizona State University, USA
Su Hsin-Hao	MIT, USA
Suomela Jukka	Aalto University, Finland
Woelfel Philipp	University of Calgary, Canada

Organizing Committee

Mikołaj Bojańczyk	University of Warsaw, Poland
Piotr Sankowski	University of Warsaw, Poland
Bartek Klin	University of Warsaw, Poland
Filip Murlak	University of Warsaw, Poland

Financial Sponsors

Microsoft
Microsoft Research
AICA
Facebook
Department of Informatics, Sapienza University of Rome
Austrian

Additional Reviewers

Abboud Amir	Abdollahi Azgomi Mohammad	Abdullah Amirali
Aceto Luca	Acharya Jayadev	Adjiashvili David
Agrawal Akanksha	Akhavi Ali	Alaei Saeed
Allender Eric	Ambainis Andris	Amir Amihood
Andoni Alexandr	Angelidakis Haris	Angelopoulos Spyros
Anshu Anurag	Antonopoulos Timos	Arroyuelo Diego
Assadi Sepehr	Atig Mohamed Faouzi	Atkey Robert
Backurs Arturs	Baillot Patrick	Balbiani Philippe
Ball Marshall	Bampas Evangelos	Bannai Hideo
Barman Siddharth	Barmpalias Georgios	Barrington David Mix
Barth Stephan	Basu Samik	Bauer Matthew
Behnezhad Soheil	Bei Xiaohui	Belovs Aleksandrs
Ben-Amram Amir	Bérczi Kristóf	Berkholz Christoph
Berman Itay	Beyersdorff Olaf	Bhaskar Umang
Bhaskara Aditya	Bienkowski Marcin	Bille Philip
Björklund Andreas	Blondin Michael	Bodini Olivier
Bodlaender Hans L.	Bodwin Greg	Bonchi Filippo
Bonnet Edouard	Boreale Michele	Borradaile Glencora
Bouland Adam	Briggs Keith	Brody Joshua
Brunet Paul	Byrka Jaroslaw	Cannon Sarah
Canonne Clément	Cao Yixin	Carayol Arnaud
Carlier Pierre	Caskurlu Bugra	Chakraborty Sourav
Chalermsook Parinya	Chang Hsien-Chih	Chang Yi-Jun
Chaplick Steven	Charron-Bost Bernadette	Chen Hubie
Chen Sitan	Chen Yi-Hsiu	Chen Yijia
Chen Yilei	Chen Yu-Fang	Chestnut Stephen
Chiplunkar Ashish	Chistikov Dmitry	Chitnis Rajesh
Chlamtac Eden	Choffrut Christian	Choudhary Keerti
Cicalese Ferdinando	Clemente Lorenzo	Cohen Alon
Cohen Ilan	Coja-Oghlan Amin	Colella Feliciano

Collet Simon
Crespi Reghizzi Stefano
Cygan Marek
Dahlgaard Søren
Daruki Samira
Das Syamantak
de Laat David
Degwekar Akshay
Della Monica Dario
Dima Catalin
Doty David
Duetting Paul
Eisner Cindy
Elkind Edith
Epping Michael
Függer Matthias
Faella Marco
Fearnley John
Feldmann Michael
Fiat Amos
Filiot Emmanuel
Fiorini Samuel
Fotakis Dimitris
Freedman Ofer
Garg Ankit
Gawrychowski Pawel
Gentilini Raffaella
Gkatzelis Vasilis
Goldner Kira
Golovach Petr
Goranci Gramoz
Grandoni Fabrizio
Grier Daniel
Guha Shibashis
Guinard Breuc
Gupta Manoj
Gutin Gregory
Høyer Peter
Hallgren Sean
Hansen Thomas Dueholm
Harvey Nick
He Meng
Hlineny Petr
Hoefer Martin
Holmgren Justin
Huang Dawei
Huang Zengfeng
I Tomohiro

Cormode Graham
Cui Shawn
Czerwiński Wojciech
Dantchev Stefan
Das Bireswar
Daviaud Laure
De Nivelle Hans
Dehghani Sina
Deshpande Amit
Dinitz Michael
Doyen Laurent
Dürr Christoph
Eldar Lior
Elmasry Amr
Epstein Leah
Gaboardi Marco
Fakcharooenphol Jittat
Fekete Sándor
Felmdan Dan
Fichtenberger Hendrik
Finkbeiner Bernd
Fischer Johannes
Fox Kyle
Friedrichs Stephan
Gasieniec Leszek
Geeraerts Gilles
Giannopoulou Archontia
Glasser Christian
Goldwurm Massimiliano
Golub Benjamin
Gordon Spencer
Gravin Nikolai
Grohe Martin
Guillon Bruno
Guo Heng
Gurjar Rohit
Haase Christoph
Hsu Justin
Han Xin
Haramaty Elad
Harwath Frederik
Hermelin Danny
Ho Hsi-Ming
Hoffmann Jan
Hoshino Naohiko
Huang Sangxia
Huang Zhiyi
Iacono John

Couteau Geoffroy
Curticapean Radu
Czyzowicz Jurek
Dartois Luc
Das Shantanu
de Brecht Matthew
De Rougemont Michel
Dell Holger
Dieudonne Yoann
Dondi Riccardo
Duchon Philippe
Dvorak Zdenek
Elias Marek
Emamjomeh-Zadeh Ehsan
Esfandiari Hossein
Ganian Robert
Fates Nazim
Feldman Moran
Fernau Henning
Fijalkow Nathanaël
Fiore Dario
Forbes Michael A.
Franceschini Gianni
Fuchsbauer Georg
Gauwin Olivier
Gelashvili Rati
Gimbert Hugo
Gmyr Robert
Göller Stefan
Gopi Sivakanth
Gouleakis Themistoklis
Gribling Sander
Grønlund Allan
Guillon Pierre
Guo Siyao
Guruganesh Guru
Haghpanah Nima
Huang Chien-Chung
Hansen Kristoffer Arnsfelt
Harsha Prahladh
Hassin Refael
Hinnenthal Kristian
Hodkinson Ian
Hofman Piotr
Houshmand Mahboobeh
Huang Shang-En
Hubacek Pavel
Ikenmeyer Christian

Immerman Neil
Inenaga Shunsuke
Iwata Yoichi
Jain Rahul
Jancar Petr
Jansen Bart M. P.
Jansen Klaus
Jeffery Stacey
Jeż Artur
Ji Zhengfeng
Jiamjitrak Wanchote
Johnson Matthew P.
Jones Mark
Jost Steffen
Jowhari Hossein
Kalaitzis Christos
Kamath Gautam
Kamath Pritish
Kammar Ohad
Kanazawa Makoto
Kannan Ravindran
Kapralov Michael
Karczmarz Adam
Kärkkäinen Juha
Katsumata Shin-Ya
Kazda Alexandr
Keiren Jeroen J.A.
Kempe David
Kesselheim Thomas
Khan Arindam
Kim Anthony
Kimelfeld Benny
King Valerie
Kini Dileep
Kiraly Tamas
Klasing Ralf
Klavzar Sandi
Kling Peter
Knudsen Mathias Bæk Tejs
Kociumaka Tomasz
Koebler Johannes
Koiran Pascal
Kolb Christina
Kolev Pavel
Komusiewicz Christian
Konrad Christian
Konur Savas
Kothari Robin
Kotrbcik Michal
Kowalik Lukasz
Kranakis Evangelos
Krasnopolsky Nadav
Kratsch Stefan
Krinninger Sebastian
Krishnaswamy Ravishankar
Krivosija Amer
Kufleitner Manfred
Kulikov Alexander
Kulkarni Janardhan
Kumar Amit
Kumar Mrinal
Kumaresan Ranjit
Künnemann Marvin
Kuusisto Antti
Kwon O-Joung
Kyng Rasmus
Löding Christof
Loff Bruno
Lohrey Markus
Laber Eduardo
Łącki Jakub
Laekhanukit Bundit
Laird James
Lang Harry
Lange Julien
Lapinskas John
Lasota SŁawomir
Laurenti Luca
Lauriere Mathieu
Le Gall Francois
Lecroq Thierry
Lee Euiwoong
Leroux Jérôme
Leverrier Anthony
Levin Asaf
Li Shi
Li Yi
Liaghat Vahid
Liao Chao
Libkin Leonid
Lin Bingkai
Lingas Andrzej
Lingxiao Huang
Liu Jingcheng
Liu Tianren
Lodaya Kamal
Lokshtanov Daniel
Longley John
Lotker Zvi
Luttik Bas
M. S. Ramanujan
Madan Vivek
Makarychev Konstantin
Mamouras Konstantinos
Mande Nikhil
Manea Florin
Manthey Bodo
Manurangsi Pasin
Mao Jieming
Mardare Radu
Marino Andrea
Markatou Evangelia Anna
Markey Nicolas
Markham Damian
Martin Barnaby
Mathieu Claire
Mathur Umang
Matuschke Jannik
Maus Yannic
McCusker Guy
McGregor Andrew
Medina Moti
Mehraban Saeed
Mehta Aranyak
Mendler Michael
Mengel Stefan
Meunier Pierre-étienne
Mezlaf David
Michaliszyn Jakub
Milius Stefan
Mimram Samuel
Misra Neeldhara
Mittal Rajat
Mnich Matthias
Mogavero Fabio
Monaco Gianpiero
Monemizadeh Morteza
Monmege Benjamin
Montanaro Ashley
Moseley Benjamin
Mozes Shay
Mucha Marcin
Mukherjee Pratyay
Munro Ian

Munteanu Alexander
Murlak Filip
Musco Cameron
Musco Christopher
Nagarajan Viswanath
Narayanan Hariharan
Natarajan Ramamoorthy Sivaramakrishnan
Nekrich Yakov
Nenadov Rajko
Newman Ilan
Nguyen Huy
Niazadeh Rad
Nicholson Patrick K.
Nichterlein André
Nielsen Jesper Sindahl
Nies Andre
Niewerth Matthias
Nigam Vivek
Norouzi Fard Ashkan
Novotný Petr
Ochremiak Joanna
Oertel Timm
Okhotin Alexander
Onak Krzysztof
Oren Sigal
Otop Jan
Otto Martin
Paes Leme Renato
Pajak Dominik
Pakusa Wied
Panangaden Prakash
Panigrahi Debmalya
Panolan Fahad
Parotsidis Nikos
Paz Ami
Peng Pan
Peng Richard
Perakis Georgia
Perifel Sylvain
Perkins Will
Perry Mor
Persiano Giuseppe
Petri Gustavo
Petrisan Daniela
Phillips Jeff
Pilipczuk MichaŁ
Plandowski Wojciech
Polonsky Andrew
Porter Timothy
Pountourakis Emmanouil
Pous Damien
Prigioniero Luca
Psomas Christos-Alexandros
Puglisi Simon
Quaas Karin
Quanrud Kent
Rabani Yuval
Rabinovich Roman
Räcke Harald
Radoszewski Jakub
Raghavan Manish
Raghothaman Mukund
Raman Rajiv
Randour Mickael
Rao Anup
Rapaport Ivan
Raskin Jean-Francois
Reidl Felix
Ren Ling
Renault Marc
Rey Anja
Reynier Pierre-Alain
Richerby David
Rika Inbal
Riveros Cristian
Rizzi Romeo
Roditty Liam
Rolínek Michal
Romashchenko Andrei
Romero Orth Miguel
Roohi Nima
Rosén Adi
Rosenbaum Will
Rothenberger Ralf
Rubinstein Aviad
Rutten Jan
Saarela Aleksi
Sadakane Kunihiko
Salcedo-Sanz Sancho
Sangnier Arnaud
Santhanam Rahul
Santocanale Luigi
Saranurak Thatchaphol
Satti Srinivasa Rao
Sau Ignasi
Saurabh Saket
Sawa Zdenek
Scagnetto Ivan
Scarlett Jonathan
Schaeffer Luke
Schmid Andreas
Schmidt Jens M.
Schmidt Ludwig
Schmidt Melanie
Schmitz Sylvain
Schöpp Ulrich
Schwartz Roy
Schweitzer Pascal
Schwiegelshohn Chris
Seddighin Saeed
Segala Roberto
Sen Pranab
Setzer Alexander
Shirmohammadi Mahsa
Siebertz Sebastian
Silva Pedro V.
Singer Yaron
Singla Sahil
Sinha Makrand
Skiena Steven
SocaŁa Arkadiusz
Sokol Dina
Sokolova Ana
Song Zhao
Sorge Manuel
Spoerhase Joachim
Srivastava Piyush
Staals Frank
Stachowiak Grzegorz
Starikovskaya Tatiana
Stefankovic Daniel
Stehle Damien
Stepanovs Igors
Stephens-Davidowitz Noah
Strothmann Thim
Sun He
Sun Nike
Svensson Ola
Talwar Kunal
Tamaki Suguru
Tan Li-Yang

Tang Bo
Tang Qiyi
Tendera Lidia
Terwijn Sebastiaan
Tirodkar Sumedh
Tiu Alwen
Tonoyan Tigran
Torán Jacobo
Toruńczyk Szymon
Totzke Patrick
Tsur Dekel
Tulsiani Madhur
Tzamos Christos
Uniyal Sumedha
Valencia Frank
van Leeuwen Erik Jan
van Stee Rob
Vardi Adi
Varma Nithin Mahendra
Vassilevska Williams Virginia
Vasudev Yadu
Vasudevan Prashant
Vaz Daniel
Velingker Ameya
Venema Yde
Venturini Rossano
Vergnaud Damien
Versari Luca
Vialette Stéphane
Vickers Steve
Viglietta Giovanni
Vinyals Marc
Vladu Adrian
Vondrak Jan
Wahlström Magnus
Wajc David
Wang Zhengyu
Wang Zihe
Ward Justin
Wellnitz Philip
Westermann Matthias
Whistler William
Widder Josef
Wieder Udi
Wiese Andreas
Wijs Anton
Will Sebastian
Winter Joost
Wlodarczyk Michal
Woeginger Gerhard J.
Wötzel Maximilian
Wrochna Marcin
Wrona MichaŁ
Wu Zhiwei Steven
Wulff-Nilsen Christian
Xia Mingji
Xu Haifeng
Xu Jiaming
Xu Shen Chen
Yakovlev Alex
Yang Dejun
Yang Kuan
Yang Lin
Yaroslavtsev Grigory
Yazdanbod Sadra
Yin Xiang
Yogev Eylon
Yoshida Yuichi
Young Neal
Yu Fang-Yi
Yu Huacheng
Yuster Raphael
Zanasi Fabio
Zandieh Amir
Zanetti Luca
Zehavi Meirav
Zeitoun Marc
Zetzsche Georg
Zeume Thomas
Zhandry Mark
Zhang Chihao
Zhang Jialin
Zhang Qin
Zhang Yumeng
Zhou Hang
Zhou Yuan
Ziegler Martin

List of Authors

Aarthi Sundaram
Centre for Quantum Technologies
Singapore
aarthims@gmail.com

Abrahamsen Mikkel
University of Copenhagen
Denmark
mikkel.abrahamsen@gmail.com

Achlioptas Dimitris
University of California Santa Cruz
USA
optas@soe.ucsc.edu

Adamczyk Marek
University of Bremen
Germany
adamczyk@dis.uniroma1.it

Agrawal Shweta
IIT Madras
India
shweta.a@gmail.com

Ahmadian Sara
University of Waterloo
Canada
sahmadian@uwaterloo.ca

Albers Susanne
TU Muenchen
Germany
albers@in.tum.de

Almagor Shaull
Department of Computer Science, Oxford University, UK
Israel
shaull.almagor@cs.ox.ac.uk

Alman Josh
MIT CSAIL
USA
jalman@mit.edu

Alstrup Stephen
University of Copenhagen
Denmark
stephen.alstrup.private@gmail.com

Alur Rajeev
University of Pennsylvania
USA
alur@cis.upenn.edu

Amarilli Antoine
LTCI, CNRS, Télécom ParisTech, Université Paris-Saclay
France
antoine.amarilli@telecom-paristech.fr

Antoniadis Antonios
Bonn University
Germany
antonios.antoniadis@mpi-inf.mpg.de

Apon Daniel
University of Maryland, College Park
USA
dapon@cs.umd.edu

Asada Kazuyuki
University of Tokyo
Japan
kzykasd+easychair@gmail.com

Ásgeirsson Eyjólfur Ingi
Reykjavik University
Iceland
eyjo@ru.is

Atserias Albert
Universitat Politecnica de Catalunya
Spain
atserias@cs.upc.edu

Babaioff Moshe
Microsoft Research
Israel
moshe@microsoft.com

Bacci Giorgio
Dept. of Computer Science, Aalborg University
Denmark
grbacci@cs.aau.dk

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Bacci Giovanni
Dept. of Computer Science, Aalborg University
Denmark
giovbacci@cs.aau.dk

Backens Miriam
University of Bristol
United Kingdom
m.backens@bristol.ac.uk

Bacquey Nicolas
INRIA Lille - CRIStAL - University of Lille
France
nicolas.bacquey@inria.fr

Baleshzar Roksana
Pennsylvania State University
USA
rxb5410@cse.psu.edu

Balle Borja
Lancaster University
United Kingdom
bballe@cs.mcgill.ca

Barbero Florian
LIRMM
France
Florian.Barbero@lirmm.fr

Barthe Gilles
IMDEA Software Institute
Spain
gjbarthe@gmail.com

Basset Nicolas
Université Libre de Bruxelles
Belgium
nicolas.basset@ulb.ac.be

Baswana Surender
I.I.T. Kanpur
India
sbaswana@cse.iitk.ac.in

Belleville Amanda
UC Davis
USA
acbelleville@ucdavis.edu

Ben Eliezer Omri
Tel Aviv University
Israel
omribene@gmail.com

Ben-Sasson Eli
Technion
Israel
eli@cs.technion.ac.il

Benedikt Michael
Oxford University
United Kingdom
Michael.Benedikt@cs.ox.ac.uk

Bernstein Aaron
TU Berlin
Germany
bernstei@gmail.com

Berthon Raphaël
ENS Rennes
France
raphael.berthon@ens-rennes.fr

Bezakova Ivona
Rochester Institute of Technology
USA
ib@cs.rit.edu

Bienkowski Marcin
Institute of Computer Science, University of Wroclaw
Poland
marcin.bienkowski@cs.uni.wroc.pl

Bilò Vittorio
University of Salento
Italy
vittorio.bilo@unisalento.it

Björklund Andreas
Lund University
Sweden
andreas.bjorklund@yahoo.se

Blelloch Guy
Computer Science Department, Carnegie Mellon University
USA
guyb@cs.cmu.edu

Blumrosen Liad
School of Business, The Hebrew University
Israel
blumrosen@gmail.com

Boczkowski Lucas
CNRS, IRIF, University Paris 7
France
lucasboczko@gmail.com

Bodwin Greg
CSAIL, MIT
USA
gbodwin@mit.edu

Bogdanov Andrej
Chinese University of Hong Kong
Hong Kong
andrejb@cse.cuhk.edu.hk

Böhnlein Toni
Universität zu Köln
Germany
boehnlein@zpr.uni-koeln.de

Boigelot Bernard
Montefiore Institute, Université de Liège
Belgium
bernard.boigelot@ulg.ac.be

Bojańczyk Mikołaj
University of Warsaw
Poland
bojan@mimuw.edu.pl

Bonnet Édouard
Middlesex University
United Kingdom
edouard.bonnet@dauphine.fr

Bourhis Pierre
CNRS CRIStAL
France
pierre.bourhis@univ-lille1.fr

Bournez Olivier
LIX & Ecole Polytechnique
France
bournez@lix.polytechnique.fr

Bozzelli Laura
University of Napoli
Italy
lr.bozzelli@gmail.com

Brandt Sebastian
ETH Zurich
Switzerland
brandts@ethz.ch

Bringmann Karl
Max Planck Institute for Informatics
Germany
kbringma@mpi-inf.mpg.de

Bury Marc
Thyssenkrupp Industrial Solutions AG
Germany
marc.bury@thyssenkrupp.com

Byrka Jaroslaw
Institute of Computer Science, University of Wroclaw
Poland
jby@cs.uni.wroc.pl

Cadilhac Michaël
WSI, Universität Tübingen
Germany
michael@cadilhac.name

Cai Leran
University of Cambridge
United Kingdom
leran.cai@cl.cam.ac.uk

Caragiannis Ioannis
University of Patras
Greece
caragian@ceid.upatras.gr

Carton Olivier
IRIF, Université Paris-Diderot
France
olivier.carton@irif.fr

Chakrabarty Deeparnab
Microsoft Research, Bangalore
India
deeparnab@gmail.com

Chalopin Jérémie
LIF, CNRS & Aix Marseille Université
France
jeremie.chalopin@lif.univ-mrs.fr

Chen Jing
Stony Brook University
USA
jingchen@cs.stonybrook.edu

Chen Shahar
Technion
Israel
shahar.chen11@gmail.com

Chepoi Victor
LIF, Aix Marseille Université & CNRS
France
chepoi@lif.univ-mrs.fr

Chiesa Alessandro
UC Berkeley
USA
alexch@berkeley.edu

Chistikov Dmitry
University of Oxford
United Kingdom
dch@mpi-sws.org

Choudhari Jayesh
IIT Gandhinagar
India
choudhari.jayesh@iitgn.ac.in

Choudhary Keerti
I.I.T. Kanpur
India
keerti@cse.iitk.ac.in

Clemente Lorenzo
University of Warsaw
Poland
clementelorenzo@gmail.com

Cleve Richard
University of Waterloo
Canada
cleve@uwaterloo.ca

Coester Christian
University of Oxford
United Kingdom
christian.coester@cs.ox.ac.uk

Cohen Ran
Tel-Aviv University
Israel
cohenran@yahoo.com

Coretti Sandro
New York University
USA
corettis@gmail.com

Curticapean Radu
Institute for Computer Science and Control of the Hungarian Academy of Sciences (MTA SZTAKI)
Hungary
radu.curticapean@gmail.com

Cygan Marek
Institute of Informatics, University of Warsaw, Poland
Poland
cygan@mimuw.edu.pl

Czerwiński Wojciech
University of Warsaw
Poland
wczerwin@mimuw.edu.pl

Dartois Luc
Université Libre de Bruxelles
Belgium
ldartois@ulb.ac.be

Dasgupta Anirban
IIT Gandhinagar
India
anirbandg@iitgn.ac.in

Datta Samir
Chennai Mathematical Institute
India
sdatta@cmi.ac.in

Daviaud Laure
University of Warsaw
Poland
ldaviaud@mimuw.edu.pl

Dehghani Sina
University of Maryland
USA
dehghani@umd.edu

Dell Holger
Saarland University and Cluster of Excellence, MMCI
Germany
holger.dell@gmail.com

Dereniowski Dariusz
Gdansk University of Technology
Poland
deren@eti.pg.gda.pl

Di Castro Dotan
Technion
Israel
dotan.dicastro@gmail.com

Diakonikolas Ilias
University of Southern California
USA
ilias.diakonikolas@gmail.com

Diekert Volker
University Stuttgart
Germany
diekert@fmi.uni-stuttgart.de

Disser Yann
TU Darmstadt
Germany
yanndisser@gmail.com

Doerr Benjamin
Max-Planck Institute for Informatics
Germany
doerr@mpi-inf.mpg.de

Dohotaru Cătălin
University of Calgary
Canada
cdohotaru@gmail.com

Döttling Nico
UC Berkeley
USA
nico.doettling@gmail.com

Doty David
UC Davis
USA
doty@ucdavis.edu

Dowek Gilles
INRIA and ENS Paris-Saclay
France
gilles.dowek@inria.fr

Eden Talya
Tel Aviv University
Israel
talyaa01@gmail.com

Ehsani Soheil
UMD
USA
soheilehsani@gmail.com

Eickmeyer Kord
TU Darmstadt
Germany
eickmeyer@mathematik.tu-darmstadt.de

Elder Murray
The University of Newcastle, Australia
Australia
murrayelder@gmail.com

Emek Yuval
Technion
Israel
yemek@ie.technion.ac.il

Espitau Thomas
Université Paris 6
France
t.espitau@gmail.com

Even Guy
Tel-Aviv University
Israel
guy@eng.tau.ac.il

Fanelli Angelo
CNRS
France
angelo.fanelli@gmail.com

Figueira Diego
LaBRI, CNRS
France
dfigueir@labri.fr

Fijalkow Nathanaël
University of Oxford
United Kingdom
nathanael.fijalkow@gmail.com

Finkel Alain
LSV, ENS Cachan, CNRS
France
finkel@lsv.fr

Flammini Michele
University of L'Aquila
Italy
michele.flammini@univaq.it

Fomin Fedor
Department of Informatics, University of Bergen
Norway
fomin@ii.uib.no

Fournier Paulin
Université de Bordeaux
France
paulin.fournier@labri.fr

Friggstad Zachary
University of Alberta
Canada
zacharyf@ualberta.ca

Gabizon Ariel
Technion
Israel
arielga@cs.technion.ac.il

Galanis Andreas
University of Oxford
United Kingdom
andreas.galanis@cs.ox.ac.uk

Galicki Alex
The University of Auckland
New Zealand
agal629@aucklanduni.ac.nz

Garay Juan
Yahoo Research
USA
garay@yahoo-inc.com

Garg Sanjam
UC Berkeley
USA
sanjamg@berkeley.edu

Gaspers Serge
University of New South Wales
Australia
sergeg@cse.unsw.edu.au

Geeraerts Gilles
Université Libre de Bruxelles
Belgium
gigeerae@ulb.ac.be

Georgiadis Loukas
University of Ioannina
Greece
loukas@gmail.com

Ghazi Badih
MIT
USA
badih@mit.edu

Giannakopoulos Yiannis
TU Munich
Germany
giannako@in.tum.de

Giannopoulou Archontia
Technische Universität Berlin
Germany
archontia.giannopoulou@gmail.com

Gimbert Hugo
CNRS, LABRI
France
hugo.gimbert@labri.fr

Gmyr Robert
Paderborn University
Germany
gmyr@mail.upb.de

Gold Omer
Tel Aviv University
Israel
omergolden@gmail.com

Goldberg Leslie Ann
University of Oxford
United Kingdom
leslie.goldberg@cs.ox.ac.uk

Goldwasser Shafi
MIT
USA
shafi@theory.csail.mit.edu

Golovach Petr
Department of Informatics, Bergen University
Norway
pgo041@uib.no

Göös Mika
Harvard
USA
mika@seas.harvard.edu

Gorain Barun
Université du Québec en Outaouais
Canada
baruniitg123@gmail.com

Gourdeau Pascale
McGill University
Canada
pascale.gourdeau@mail.mcgill.ca

Graf Daniel
ETH Zürich
Switzerland
grafdan@ethz.ch

Grandjean Etienne
GREYC, University of Caen
France
etienne.grandjean@unicaen.fr

Grandoni Fabrizio
IDSIA, USI-SUPSI, Lugano
Switzerland
fabrizio@idsia.ch

Gravin Nick
MIT
USA
ngravin@mit.edu

Groß Martin
University of Waterloo
Canada
gross@math.TU-Berlin.DE

Grossman Ofer
MIT
USA
ofer.grossman@gmail.com

Gu Yan
Carnegie Mellon University
USA
yan.gu@cs.cmu.edu

Guillon Bruno
University of Warsaw
Poland
guillon.bruno+cs@gmail.com

Guldstrand Larsen Kim
Dept. of Computer Science, Aalborg University
Denmark
kgl@cs.aau.dk

Gupta Manoj
Indian Institute of Technology, Gandhinagar
India
gmanoj@iitgn.ac.in

Guruswami Venkatesan
Carnegie Mellon University
USA
guruswami@cmu.edu

Gutin Gregory
Royal Holloway
United Kingdom
g.gutin@rhul.ac.uk

Haase Christoph
University of Oxford
United Kingdom
Christoph.Haase@cs.ox.ac.uk

Hajiaghayi Mohammadtaghi
University of Maryland, College Park
USA
hajiagha@cs.umd.edu

Halldorsson Magnus M.
Reykjavik University
Iceland
magnusmh@gmail.com

Hansen Thomas Dueholm
Aarhus University
Denmark
tdh@cs.au.dk

Henzinger Monika
Faculty of Computer Science, Universität Wien
Austria
monika.henzinger@univie.ac.at

Hinnenthal Kristian
Paderborn University
Germany
krijan@mail.upb.de

Hoefer Martin
Goethe University Frankfurt/Main
Germany
mhoefer@mpi-inf.mpg.de

Hoeksma Ruben
Universidad de Chile
Netherlands
rphoeksma@gmail.com

Holm Jacob
Department of Computer Science, University of Copenhagen
Denmark
jaho@di.ku.dk

Høyer Peter
University of Calgary
Canada
hoyer@ucalgary.ca

Hsu Justin
University of Pennsylvania
USA
email@justinh.su

Iliopoulos Fotis
University of California Berkeley
USA
fotis.iliopoulos@berkeley.edu

Italiano Giuseppe F.
University of Rome Tor Vergata
Italy
pino.italiano@gmail.com

Iwata Yoichi
National Institute of Informatics
Japan
yiwata@nii.ac.jp

Jayaram Rajesh
Brown University
USA
rajesh_jayaram@brown.edu

Jayram T.S.
IBM Almaden
USA
jayram@us.ibm.com

Jecker Ismaël
Université libre de Bruxelles
Belgium
ismael.jecker@gmail.com

Jeż Artur
University of Wroclaw, Institute of Computer Science
Poland
aje@cs.uni.wroc.pl

Jung Jean Christoph
Universität Bremen
Germany
jeanjung@informatik.uni-bremen.de

Kane Daniel
University of California, San Diego
USA
aladkeenin@gmail.com

Kärkkäinen Juha
University of Helsinki
Finland
juha.karkkainen@cs.helsinki.fi

Karnin Zohar
Yahoo Labs
USA
zkarnin@gmail.com

Kaski Petteri
Department of Computer Science, Aalto University, Helsinki, Finland
Finland
petteri.kaski@aalto.fi

Kelmendi Edon
LaBRI
France
edon.kelmendi@labri.Fr

Kerenidis Iordanis
CNRS, IRIF, University Paris 7
France
iordanis.kerenidis@irif.fr

Khan Shahbaz
Indian Institute of Technology, Kanpur
India
amus_hawk@yahoo.co.in

Kindler Guy
The Weizmann Institute of Science
Israel
guy.kindler@weizmann.ac.il

Klin Bartek
University of Warsaw
Poland
klin@mimuw.edu.pl

Knudsen Mathias Bæk Tejs
University of Copenhagen
Denmark
mathias@tejs.dk

Kobayashi Naoki
University of Tokyo
Japan
koba@is.s.u-tokyo.ac.jp

Kodric Bojana
Max-Planck-Institut für Informatik
Germany
bojana@mpi-inf.mpg.de

Kohler Matthias
Technical University of Munich
Germany
kohler@in.tum.de

Korman Amos
CNRS and Université Paris Diderot - Paris 7
France
amos.korman@gmail.com

Korman Simon
Weizmann Institute of Science
Israel
simon.korman@gmail.com

Kosowski Adrian
Inria and IRIF, Paris
France
adrian.kosowski@inria.fr

Kostrygin Anatolii
Ecole Polytechnique
France
anatolii.kostrygin@gmail.com

Koutis Ioannis
Department of Computer Science, University of Puerto Rico – Rio Piedras
Puerto Rico
i.koutis@gmail.com

Koutsoupias Elias
University of Oxford
United Kingdom
elias@cs.ox.ac.uk

Kraft Dennis
TU Muenchen
Germany
dennis.kraft@in.tum.de

Kratsch Stefan
University of Bonn
Germany
kratsch@cs.uni-bonn.de

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

Kreutzer Stephan
TU Berlin
Germany
stephan.kreutzer@tu-berlin.de

Krinninger Sebastian
University of Vienna
Austria
sebastian.krinninger@univie.ac.at

Künnemann Marvin
University of California, San Diego
USA
mkuennemann@eng.ucsd.edu

Kwon O-Joung
TU Berlin
Germany
o.kwon@tu-berlin.de

Lambilliotte Antonin
École Normale Supérieure de Lyon
France
antonin.lambilliotte@ens-lyon.fr

Lasota Sławomir
Warsaw University
Poland
sl@mimuw.edu.pl

Lazic Ranko
University of Warwick
United Kingdom
R.S.Lazic@warwick.ac.uk

Lazos Philip
University of Oxford
United Kingdom
filzos@cs.ox.ac.uk

Lee Edward J.
UNSW / Data61
Australia
edward.jay.lee@gmail.com

Lee Euiwoong
Carnegie Mellon University
USA
euiwoonl@cs.cmu.edu

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

Leroux Jerome
LaBRI, CNRS
France
jerome.leroux@labri.fr

Levi Reut
MPI
Germany
reuti.levi@googlemail.com

Lewin-Eytan Liane
Yahoo Labs
Israel
liane@yahoo-inc.com

Lhote Nathan
Université de Bordeaux
France
nlhote@labri.fr

Li Bo
Stony Brook University
USA
boli2@cs.stonybrook.edu

Li Yi
Nanyang Technological University
Singapore
leeyi@umich.edu

Li Yingkai
Stony Brook University
USA
yingkli@cs.stonybrook.edu

Lin Chengyu
Columbia University
USA
chengyu@cs.columbia.edu

Lin Jiabao
Peking University
China
joblin@pku.edu.cn

Linhares Andre
University of Waterloo
Canada
andre.linhares@gmail.com

Lokshtanov Daniel
UiB
Norway
daniello@ii.uib.no

Louis Jachiet
Université Grenoble Alpes
France
louis.jachiet@inria.fr

Lozes Etienne
LSV, ENS Cachan, CNRS
France
lozes@lsv.ens-cachan.fr

Lutz Carsten
Universität Bremen
Germany
clu@informatik.uni-bremen.de

M. S. Ramanujan
TU Wien
India
msramanujan@gmail.com

Magniez Frédéric
CNRS, IRIF, University Paris 7
France
magniez@cnrs.fr

Mai Tung
Georgia Institute of Technology
USA
Maithanhtung89@gmail.com

Mainz Isabelle
Montefiore Institute, Université de Liège
Belgium
isabelle.mainz@ulg.ac.be

Mamouras Konstantinos
University of Pennsylvania
USA
kmamouras@gmail.com

Mancinska Laura
University of Bristol
United Kingdom
laura.mancinska@gmail.com

Manurangsi Pasin
University of California, Berkeley
USA
pasin@berkeley.edu

Mardare Radu
Dept. of Computer Science, Aalborg University
Denmark
mardare@cs.aau.dk

Marsault Victor
Department of Mathematics, Université de Liège
Belgium
victor.marsault@ulg.ac.be

Martel Mauricio
Universität Bremen
Germany
mauricio.martel@gmail.com

Matuschke Jannik
TU Berlin
Germany
matuschke@math.tu-berlin.de

Mazowiecki Filip
University of Warwick
United Kingdom
F.Mazowiecki@warwick.ac.uk

Mccormick Tom
Sauder School of Business, UBC
Canada
tom.mccormick@sauder.ubc.ca

Medina Moti
MPI
Germany
moti.medina@gmail.com

Meissner Julie
TU Berlin MA Sek. 5-2
Germany
jmeiss@math.tu-berlin.de

Michail Othon
Department of Computer Science, University of Liverpool, UK
United Kingdom
Othon.Michail@liverpool.ac.uk

Misra Neeldhara
Indian Institute of Science
India
mail@neeldhara.com

Mnich Matthias
Universität Bonn
Germany
mmnich@uni-bonn.de

Molinari Alberto
University of Udine
Italy
molinari.alberto@gmail.com

Monaco Gianpiero
DISIM. University of L'Aquila
Italy
gianpiero.monaco@di.univaq.it

Monemizadeh Morteza
Rutgers University
USA
mortezam@dimacs.rutgers.edu

Montanari Angelo
University of Udine
Italy
angelo.montanari@uniud.it

Mouawad Amer
University of Bergen
Norway
amer.mouawad@gmail.com

Mucha Marcin
Institute of Informatics, University of Warsaw, Poland
Poland
mucha@mimuw.edu.pl

Mukherjee Anish
Chennai Mathematical Institute
India
anish@cmi.ac.in

Mukherjee Pratyay
UC Berkeley
USA
pratyay85@gmail.com

Muthukrishnan S.
Rutgers University
USA
muthu@cs.rutgers.edu

Nagarajan Viswanath
University of Michigan
USA
viswa@umich.edu

Nakos Vasileios
Harvard University
USA
vasileiosnakos@g.harvard.edu

Naor Seffi
Computer Science Dept., Technion, Haifa, Israel
Israel
naor@cs.technion.ac.il

Nikishkin Vladimir
University of Edinburgh
United Kingdom
v.nikishkin@sms.ed.ac.uk

Nisan Noam
Microsoft Reserach and Hebrew University
Israel
noam@cs.huji.ac.il

O'Donnell Ryan
Carnegie Mellon University
USA
odonnell@cs.cmu.edu

Ochremiak Joanna
Universite de Paris VII
France
ochremiak@mimuw.edu.pl

Olive Frédéric
LIF, Aix-Marseille Université
France
Frederic.Olive@lif.univ-mrs.fr

Oriolo Gianpaolo
Universita di Roma "Tor Vergata"
Italy
oriolo@disp.uniroma2.it

Ouaknine Joel
Department of Computer Science, Oxford University, UK
United Kingdom
joel@cs.ox.ac.uk

Pallavoor Ramesh Krishnan S.
Pennsylvania State University
USA
ramesh@psu.edu

Panageas Ioannis
MIT and SUTD
USA
panageasj@gmail.com

Panangaden Prakash
McGill University
Canada
prakash@cs.mcgill.ca

Panolan Fahad
Department of Informatics, University of Bergen, Norway
Norway
fahad.panolan@ii.uib.no

Paperman Charles
WSI, Universität Tübingen
Germany
charles.paperman@gmail.com

Parotsidis Nikos
University of Rome Tor Vergata
Italy
nickparo1@gmail.com

Parter Merav
CSAIL, MIT
USA
parter@mit.edu

Paturi Ramamohan
University of California, San Diego
USA
paturi@cs.ucsd.edu

Paul Christophe
CNRS - LIRMM
France
paul@lirmm.fr

Pelc Andrzej
Université du Québec en Outaouais
Canada
Andrzej.Pelc@uqo.ca

Penelle Vincent
University of Warsaw
Poland
penelle@univ-mlv.fr

Peng Pan
Faculty of Computer Science, University of Vienna
Austria
pan.peng@univie.ac.at

Peres Yuval
Microsoft Research
USA
peres@microsoft.com

Peron Adriano
University of Napoli
Italy
adrperon@unina.it

Piątkowski Marcin
Nicolaus Copernicus University, Toruń,
Poland
Poland
martinp@mat.uni.torun.pl

Pietrzak Krzysztof
IST Austria
Austria
krzpie@gmail.com

Pilipczuk Michał
University of Warsaw
Poland
michal.pilipczuk@mimuw.edu.pl

Pitassi Toniann
University of Toronto
Canada
toni@cs.toronto.edu

Pouly Amaury
LIX & FCT
France
amaury.pouly@gmail.com

Preet Singh Ishaan
IIT Delhi
India
ishaanps92@gmail.com

Price Eric
The University of Texas at Austin
USA
ecprice@cs.utexas.edu

Pudlák Pavel
The Czech Academy of Sciences
Czech Republic
pudlak@math.cas.cz

Puglisi Simon
University of Helsinki
Finland
simon.j.puglisi@gmail.com

Rabinovich Roman
TU Berlin
Germany
roman.rabinovich@tu-berlin.de

Räcke Harald
Technical University of Munich
Germany
raecke@in.tum.de

Raghavendra Prasad
UC Berkeley
USA
raghavendra@berkeley.edu

Randour Mickael
ULB - Université libre de Bruxelles
Belgium
mickael.randour@gmail.com

Raskhodnikova Sofya
Pennsylvania State University
USA
sofya@cse.psu.edu

Raskin Jean-Francois
Université Libre de Bruxelles (U.L.B.)
Belgium
jraskin@ulb.ac.be

Raskin Mikhail
Aarhus University
Denmark
raskin@mccme.ru

Raymond Jean-Florent
MIMUW, University of Warsaw and
LIRMM, University Montpellier
France
jean-florent.raymond@mimuw.edu.pl

Reichman Daniel
University of California, Berkeley
Israel
daniel.reichman@gmail.com

Reidl Felix
NC State
USA
felix.reidl@gmail.com

Reiter Fabian
IRIF, Université Paris Diderot
France
fabian.reiter@gmail.com

Riabzev Michael
Technion
Israel
mriabzev@cs.technion.ac.il

Rigo Michel
Department of Mathematics, Université de Liège
Belgium
m.rigo@ulg.ac.be

Roberson David
University College London
United Kingdom
davideroberson@gmail.com

Rodeh Yoav
Weizmann Institute
Israel
yoav.rodeh@gmail.com

Roditty Liam
Bar-Ilan University
Israel
liam.roditty@gmail.com

Ron Dana
Tel Aviv University
Israel
danaron@tau.ac.il

Rosén Adi
CNRS and Université Paris Diderot
France
adiro@liafa.univ-paris-diderot.fr

Rossman Benjamin
University of Toronto
Canada
rossman@cs.toronto.edu

Ronitt Rubinfeld
MIT and Tel Aviv University
USA and Israel
ronitt@csail.mit.edu

Rubinstein Aviad
UC Berkeley
USA
aviad@eecs.berkeley.edu

Rümmele Stefan
University of Sydney
Australia
stefan.rummele@sydney.edu.au

Saffidine Abdallah
University of New South Wales
Australia
abdallah.saffidine@gmail.com

Saha Barna
University of Massachussets, Amherst
USA
Barna@cs.umass.edu

Sala Pietro
University of Verona
Italy
pietro.sala@univr.it

Samal Robert
Charles University
Czech Republic
samal@iuuk.mff.cuni.cz

Sankur Ocan
CNRS
France
ocan.sankur@irisa.fr

Sato Tetsuya
Research Institute for Mathematical Sciences, Kyoto University
Japan
satoutet@kurims.kyoto-u.ac.jp

Sauerwald Thomas
University of Cambridge
United Kingdom
thomas.sauerwald@cl.cam.ac.uk

Saurabh Saket
The Institute of Mathematical Sciences, Chennai
India
saket@imsc.res.in

Schaudt Oliver
University of Cologne
Germany
schaudto@uni-koeln.de

Scheder Dominik
Shanghai Jiaotong University
China
dominik@cs.sjtu.edu.cn

Scheideler Christian
Paderborn University
Germany
scheideler@mail.upb.de

Schlipf Lena
FernUniversität in Hagen
Germany
lena.schlipf@fernuni-hagen.de

Schmidt Jens M.
TU Ilmenau
Germany
jens.schmidt@tu-ilmenau.de

Schneider Stefan
University of California, San Diego
USA
stschnei@cs.ucsd.edu

Schneider Thomas
University of Bremen
Germany
tschneider@informatik.uni-bremen.de

Schwartz Roy
Technion
Israel
schwartz.roi@gmail.com

Schweitzer Pascal
RWTH Aachen University
Germany
schweitzer@informatik.rwth-aachen.de

Schwentick Thomas
TU Dortmund University
Germany
thomas.schwentick@udo.edu

Schwiegelshohn Chris
TU Dortmund
Germany
chris.schwiegelshohn@tu-dortmund.de

Seddighin Saeed
University of Maryland
USA
saeedreza.seddighin@gmail.com

Seshadhri C.
University of California, Santa Cruz
USA
sesh@ucsc.edu

Severini Simone
University College London
United Kingdom
simoseve@gmail.com

Sharir Micha
Tel-Aviv University, School of Computer Science, and Courant Institute of Mathematical Science, New York University
Israel
michas@post.tau.ac.il

Shen Xiangkun
University of Michigan
USA
xkshen@umich.edu

Siebertz Sebastian
University of Warsaw
Poland
siebertz@mimuw.edu.pl

Sivan Balasubramanian
Google Research, New York
USA
balusivan@google.com

Skorski Maciej
IST Austria
Poland
maciej.skorski@gmail.com

Skretas George
Computer Engineering and Informatics Department (CEID), University of Patras, Greece
Greece
skretasg8@gmail.com

Sohler Christian
TU Dortmund
Germany
christian.sohler@tu-dortmund.de

Soloveichik David
University of Texas at Austin
USA
david.soloveichik@utexas.edu

Song Zhao
The University of Texas at Austin
USA
zhaos@utexas.edu

Spirakis Paul
University of Liverpool, Computer Technology Institute, and University of Patras
UK and Greece
P.Spirakis@liverpool.ac.uk

Spooner Nicholas
University of Toronto
Canada
spooner@cs.toronto.edu

Srinivasan Srikanth
IIT Bombay
India
srinivasan.srikanth@gmail.com

Stanford Caleb
University of Pennsylvania
USA
castan@cis.upenn.edu

Stefan Mengel
CNRS, CRIL UMR 8188; France
France
mengel@cril.fr

Stefankovic Daniel
University of Rochester
USA
stefanko@cs.rochester.edu

Stöckel Morten
University of Copenhagen
Denmark
morten.stockel@gmail.com

Strub Pierre-Yves
École Polytechnique - Paris Saclay
Spain
pierre-yves@strub.nu

Sudan Madhu
Harvard University
USA
madhu@cs.harvard.edu

Sun Yihan
Carnegie Mellon University
USA
yihans@cs.cmu.edu

Sutre Gregoire
LaBRI, CNRS
France
sutre@labri.fr

Swamy Chaitanya
University of Waterloo
Canada
cswamy@uwaterloo.ca

Talebanfard Navid
The Czech Academy of Sciences
Czech Republic
talebanfard@math.cas.cz

Thilikos Dimitrios
Department of Mathematics, National and Kapodistrian University of Athens
Greece
sedthilk@thilikos.info

Thorup Mikkel
Department of Computer Science, University of Copenhagen
Denmark
mikkel2thorup@gmail.com

Tonoyan Tigran
Reykjavik University
Iceland
ttonoyan@gmail.com

Trabelsi Ohad
Weizmann Institute of Science, Israel
Israel
ohad_trabelsi@yahoo.com

Uitto Jara
University of Freiburg
Germany
jara.uitto@gmail.com

Uznański Przemysław
ETH Zürich
Switzerland
izulin@gmail.com

Vanden Boom Michael
Department of Computer Science, University of Oxford
United Kingdom
michael.vandenboom@cs.ox.ac.uk

Varvitsiotis Antonios
Centre for Quantum Technologies and Nanyang Technological University
Singapore
avarvits@gmail.com

Vassilevska Williams Virginia
CSAIL, MIT
USA
virgi@mit.edu

Vazirani Vijay
Georgia Tech
USA
vazirani@cc.gatech.edu

Verschae José
Pontifical Catholic University of Chile
Chile
jverscha@gmail.com

Vlassis Nikos
Adobe Inc
USA
vlassis@adobe.com

Vortmeier Nils
TU Dortmund University
Germany
nils.vortmeier@uni-dortmund.de

Wahlström Magnus
Royal Holloway
United Kingdom
Magnus.Wahlstrom@rhul.ac.uk

Wang Chunhao
University of Waterloo
Canada
c265wang@uwaterloo.ca

Wang Hanpin
Peking University
China
whpxhy@pku.edu.cn

Watson Thomas
University of Memphis
USA
Thomas.Watson@memphis.edu

Wattenhofer Roger
ETH Zurich
Switzerland
wattenhofer@ethz.ch

Węgrzycki Karol
Institute of Informatics, University of Warsaw, Poland
Poland
k.wegrzycki@mimuw.edu.pl

Weitz Benjamin
UC Berkeley
USA
bsweitz123@gmail.com

Wiese Andreas
Max Planck Institute for Informatics
Germany
awiese@mpi-inf.mpg.de

Williamson Christopher
Chinese University of Hong Kong
Hong Kong
chris@cse.cuhk.edu.hk

Włodarczyk Michał
University of Warsaw
Poland
m.wlodarczyk@mimuw.edu.pl

Wolter Frank
University of Liverpool
United Kingdom
wolter@liverpool.ac.uk

Woodruff David P.
IBM Almaden
USA
dpwoodru@us.ibm.com

Worrell James
Department of Computer Science, Oxford University, UK
United Kingdom
jbw@cs.ox.ac.uk

Wrochna Marcin
University of Warsaw
Poland
m.wrochna@mimuw.edu.pl

Xing Chaoping
Nanyang Technological Univeristy
Singapore
xingcp@ntu.edu.sg

Yang Kuan
University of Oxford
United Kingdom
kuan.yang.6@gmail.com

Yuan Chen
Nanyang Technological Univeristy
Singapore
yuan0064@e.ntu.edu.sg

Zehavi Meirav
University of Bergen
Norway
zehavimeirav@gmail.com

Zeume Thomas
TU Dortmund University
Germany
thomas.zeume@cs.tu-dortmund.de

Zhang Shengyu
Chinese University of Hong Kong
Hong Kong
syzhang@cse.cuhk.edu.hk

Zikas Vassilis
RPI
USA
vzikas@cs.rpi.edu

Zou Mengchuan
Inria and IRIF, Paris
France
mengchuan.zou@inria.fr

Orbit-Finite Sets and Their Algorithms*†

Mikołaj Bojańczyk

University of Warsaw, Warsaw, Poland
bojan@mimuw.edu.pl

Abstract

An introduction to *orbit-finite sets*, which are a type of sets that are infinite enough to describe interesting examples, and finite enough to have algorithms running on them. The notion of orbit-finiteness is illustrated on the example of register automata, an automaton model dealing with infinite alphabets.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Orbit-finite sets, sets with atoms, data words, register automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.1

Category Invited Talk

1 Introduction

An orbit-finite set is a set that is constructed using some infinite logical structure, such as $(\mathbb{N}, =)$ or $(\mathbb{Q}, <)$, and which is finite up to automorphisms of that structure. For example, if the structure is $(\mathbb{N}, =)$, then

$$\{X : X \subseteq \mathbb{N} \text{ and } |X| \leq 3\}$$

is an orbit-finite set, because automorphisms of the structure (i.e. permutations of $\mathbb{N}$) can be used to map any subset $X \subseteq \mathbb{N}$ to any other subset of same cardinality, and therefore the set has four elements (cardinalities 0, 1, 2, 3) up to automorphisms.

The goal of this paper is to give a gentle introduction to orbit-finite sets, in particular to explain how they can be represented and manipulated using algorithms. As a running example we use register automata over infinite alphabets. A more detailed description can be found in the lecture notes [5].

2 The running example: register automata

As our running example for describing orbit-finite sets, we consider register automata over data words, and their associated decision problems such as emptiness or minimisation.

Typically, formal language theory uses finite alphabets. Here is an example which uses an infinite alphabet.

▶ **Running Example.** *Let $\mathbb{A}$ be some infinite set. As our running example, consider the language*

$$\{w \in \mathbb{A}^* : \textit{ at most two distinct letters appear in } w\}. \tag{1}$$

* This paper is part of LIPA, a project funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 683080).
† I would like to thank Thomas Colcombet and Szymon Toruńczyk for comments on a first draft.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 1; pp. 1:1–1:14

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

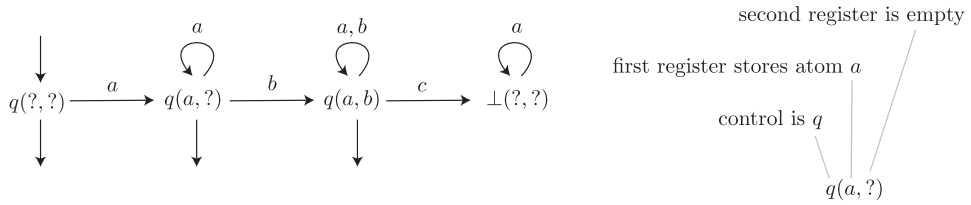


Figure 1 There are two possible values for the control state, q and $\bot$. A configuration of the automaton consists of a value for the control plus a valuation of the registers. Dangling arrows indicate initial and accepting configurations. The variables a, b, c range over distinct atoms, i.e. to get the automaton one should instantiate the picture for every triple $(a, b, c) \in \mathbb{A}^3$ of distinct atoms.

In our running example, the letters are only compared with respect to equality. Words as in the example are called languages of *data words*. The most common type of alphabet for data words is of the form $\Sigma \times \mathbb{A}$, where Σ is a finite set and $\mathbb{A}$ is an infinite set whose elements can only be compared for equality. We will use the name *atoms* for $\mathbb{A}$; to underline that they have no structure except for equality. Automata that process data words (and more complicated objects, such as data trees) are a popular model in database theory (e.g. an XML document is conveniently described as a type of data tree) or in the theory of verification. See the survey [25] for more on such automata.

One of the most basic automata models for data words is register automata (introduced in [17] under the name of *finite memory automata*). This is a type of automaton which uses finitely many registers to store some of the atoms that have been seen so far. We will use register automata as our running example of orbit-finite sets.

▶ **Running Example.** *This language in the running example, namely "at most two letters (atoms) appear" is recognised by a register automaton with two registers. The registers are used to store the at most two distinct atoms in the input. Once the two registers are filled up and a fresh third atom appears, the automaton enters a rejecting sink state. The automaton is shown in Figure 1.*

In general, the space of configurations of a register automaton is defined by giving a finite set of control states and a set of register names. A configuration is then a pair: (control state, partial function from register names to atoms). For the precise syntax of the transition relation (and notions of initial and final configurations), we refer to [17]; for the purposes of this paper it suffices to say that the syntax is designed so that transitions depend on the atoms in a way which only uses equality. Register automata, and especially deterministic register automata, are one of the simplest automaton models for data words, e.g. they are not expressive enough to recognise the language "all input letters are different". For more expressive models, see [25].

3 Mild extensions of register automata

To motivate the introduction of orbit-finite and definable sets, which are the topic of this paper, we present three mild requirements for extending the model of register automata. We will then show that these requirements can be met by automata with definable (or equivalently, orbit-finite) descriptions.

3.1 Minimisation

One natural thing to do with (deterministic) register automata is to minimise them. As we will see, the register mechanism is not well suited to this task.

▶ **Running Example.** *Consider the automaton from Figure 1, which uses two registers to recognise words with at most two distinct atoms. This automaton has different configurations after reading inputs ab and ba, because its configuration implicitly remembers which letter was read first. A minimal automaton, on the other hand, should have the same configuration after reading these two inputs, because the language is commutative. In a minimal automaton, the set of reachable configurations should be something like:*

$$\underbrace{\{\epsilon, \bot\}}_{\text{initial state and rejecting sink}} \cup \underbrace{\{a : a \in \mathbb{A}\}}_{\text{words that have exactly one atom}} \cup \underbrace{\{\{a, b\} : a \neq b \in \mathbb{A}\}}_{\text{word that have exactly two atoms}} .$$

Such a configuration set cannot be achieved with the register mechanism.

A workaround for the problems described above would be to allow registers which store unordered sets of atoms. Here is another example language, where the workaround with unordered sets of atoms fails.

▶ **Example 1.** Consider the following language

$$\{wv : w, v \in \mathbb{A}^3 \text{ are equal up to cyclic shift}\}. \tag{2}$$

For this language, the set of configurations of a minimal automaton reachable after reading three letters should be

$$\{ \underbrace{\{abc, bca, cab\}}_{\substack{\text{equivalence class of a} \\ \text{three letter word up} \\ \text{to cyclic shifts}}} : a, b, c \in \mathbb{A}\}.$$

Example 1 and the running example essentially exhaust the possible problems with miminisation: to minimise register automata it suffices to consider a model where each control state has a varying number of registers, and there might be some group acting on the registers (e.g. so that the registers form an unordered set, or can be shifted cyclically, etc.), see [8, Section 6]. The idea to use groups acting on registers dates back to [23].

Why is it so important to minimise automata? A minimal automaton is not just small – with the obvious efficiency advantages – but more importantly it is a canonical representation of its recognised language.

One use for canonical automata is that for the correctness of certain algorithms, it is useful to assume that the input is a canonical automaton. An example is learning algorithms, whose running time bounds and correctness proofs are based on minimal automata, see e.g. [20] for a variant of the Angluin algorithm for register automata. Another example is algorithms which input an automaton and decide if its recognised language is definable in some logic, see [2, 6]; here non-definability is typically equivalent to some kind of forbidden pattern in the canonical automaton.

Another use for canonical automata is that they can be useful when trying to better understand "regularity" for languages over infinite an alphabet. There is a multitude of automata models for infinite alphabets, see [22, 25], most of them with different expressive powers, and one naturally asks [4]: which model defines the "regular languages"? Over finite

alphabets, the Myhill-Nerode theorem gives a convincing machine independent characterisation of regular languages in terms of minimisation: a language is regular if and only if it has finitely many equivalence classes in their syntactic congruence (and these equivalence classes form the states of the canonical minimal automaton). Therefore, a natural idea is to study the syntactic congruences of languages such as the one in the running example, and try to find devices which store the information from the syntactic congruence and nothing else. This idea was pursued for monoids (corresponding to a two-sided syntactic congruence) in [6], for deterministic automata (corresponding to a one-sided syntactic congruence, which is different from the two-side one in the presence of infinite alphabets) in [8], and for deterministic timed automata in [3, 10]. In all of these cases the straightforward register mechanism (without group actions and other features) is insufficient to allow minimisation. See also [21] for algorithmic aspects of minimising register automata.

3.2 More general input alphabets

In data words and register automata, the input alphabet is typically assumed to be of the form $\Sigma \times \mathbb{A}$ for some finite set Σ. Why not allow slightly more general input alphabets, such as pairs of atoms $\mathbb{A}^2$ or unordered pairs of atoms $\binom{\mathbb{A}}{2}$? At first sight this seems like a needless generalisation, but it turns out that some interesting theoretical issues appear only for the more general alphabets. As an example [18, Example 2.5], which admittedly goes far beyond register automata because it uses Turing machines, consider the following set:

$$\{\{\{a, b, c\}, \{d, e, f\}\} : a, b, c, d, e, f \in \mathbb{A} \text{ are pairwise different}\}.$$

Suppose that the input alphabet Σ is the above set, and consider the language:

$$\{wv : w, v \in \Sigma^* \text{ are such that } \pi(w) = v \text{ for some permutation of the atoms } \pi\}.$$

In [18] it is shown that the above language witnesses that deterministic and nondeterministic Turing machines (in a suitable generalisation for infinite alphabets using atoms) have different expressive powers, and simpler alphabets (e.g. alphabets which talk about less than six atoms) do not witness this. This result builds on [9], which in turn builds on the seminal Cai-Fürer-Immermann [27] construction. The readers familiar with [27] will recognise the use of six atoms in the alphabet Σ.

3.3 Atoms with more structure than just equality

In our definition of data words and register automata, the atoms $\mathbb{A}$ had equality as the only available structure. Why not allow for more structure? Data words with additional structure, such as a total order, have long been present in the literature on data words. For example, [13] shows that emptiness is decidable for register automata where the input alphabet is $(\mathbb{N}, <)$ and the register operations can compare letters with respect to the order. Another important example is timed automata [1], which can be viewed as a special kind of register automata where the atoms are $(\mathbb{Q}, <, +1)$, see [10] and [14, 15] for the case of pushdown automata. Other examples come from modelling programs interacting with a database, see e.g. [26, 11]; in these applications the atoms might have e.g. an arbitrary graph structure.

4 Definable sets

In Section 2 we described register automata, and in Section 3 we discussed three requirements for a more general model: it should minimise (Section 3.1); it should allow more general

input alphabets than only the atoms (Section 3.2); and it should allow more structure on the atoms than just equality (Section 3.3). In this section we present such a model, using the notion of definable sets. The general idea is that definable sets are those that can be constructed using set-builder notation. Before giving the precise definition, consider some examples.

▶ **Example 2.** Here are some of the sets that we have seen so far:

atoms	$\mathbb{A}$
ordered pairs of atoms	$\mathbb{A}^2$
unordered pairs of atoms	$\{\{a,b\} : a,b \in \mathbb{A}\}$
states in the minimal automaton from the running example	$\{\epsilon, \bot\} \cup \{a : a \in \mathbb{A}\} \cup \{\{a,b\} : a \neq b \in \mathbb{A}\}$
triples of atoms modulo cyclic shift	$\{\{abc, bca, cab\} : a,b,c \in \mathbb{A}\}$
input alphabet for a language which witnesses that Turing machines with atoms do not determinise	$\{\{\{a,b,c\},\{d,e,f\} :$ $a,b,c,d,e,f \in \mathbb{A}$ are distinct$\}$

The above examples used only equality. In the spirit of Section 3.3, let us consider some examples which assume that the atoms are equipped with structure other than equality.

▶ **Example 3.** Assume that the atoms are equipped with a total order, and assume that 5 is one of the atoms. Here are some examples of sets defined using set builder notation:

atoms smaller than 5	$\{a : a \in \mathbb{A} \text{ with } a < 5\}$
all closed intervals	$\{\{c : c \in \mathbb{A} \text{ with } a \leq c \leq b\} : a,b \in \mathbb{A} \text{ with } a < b\}$

We now give a more formal description of set-builder notation and sets defined by it. A parameter is an underlying logical structure $\mathbb{A}$, i.e. a universe equipped with some relations and functions (equality is for free), which will be referred to as the *atoms*.

▶ **Example 4.** Here are some examples of logical structures that we will use as atoms:

$$\underbrace{\mathbb{A} = (\mathbb{N}, =)}_{\text{equality only}} \qquad \underbrace{\mathbb{A} = (\mathbb{Q}, <)}_{\text{ordered rationals}} \qquad \underbrace{\mathbb{A} = (\mathbb{N}, +)}_{\text{Presburger arithmetic}} \qquad \underbrace{\mathbb{A} = (\mathbb{N}, +, \times)}_{\text{arithmetic}}$$

In the first structure, we write the equality symbol (which is implicitly assumed to be always available) just to underline that it is the only structure available.

Given a logical structure $\mathbb{A}$, we define *set-builder expressions over* $\mathbb{A}$ by induction as follows. Fix some infinite set of variables $\{a, b, c, \ldots\}$, which will be used in the set-builder expressions, and which are intended to range over atoms, i.e. elements of the universe of $\mathbb{A}$. Subexpressions in set-builder expression can have free variables, but in the end we are interested in an outermost expression with no free variables. There are four constructions: each element of $\mathbb{A}$ is an expression (constant expression); each variable is an expression (variable expression); and expressions can be combined using binary union and set comprehension. These constructions are illustrated in Figure 2.

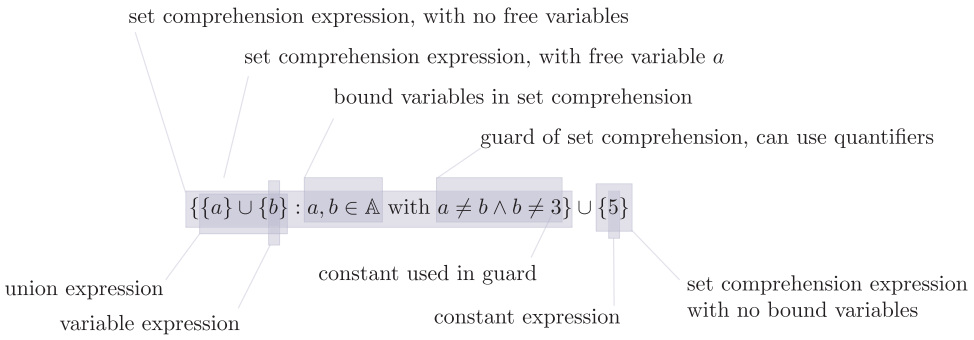


Figure 2 A set-builder expression over $\mathbb{A} = (\mathbb{N}, =)$.

The semantics of a set-builder expression is a function which inputs a valuation of its free variables (i.e. a function from the free variables to the universe of $\mathbb{A}$) and outputs an atom, a set, a set of sets, etc. defined in the obvious way. A *definable set over* $\mathbb{A}$ is the semantics of a set-builder expression without free variables. The reader will readily see that all sets mentioned in Example 2 are definable regardless of the choice of $\mathbb{A}$ (see the running example below for an explanation of how pairing and elements such as $\bot$ are encoded) and the sets mentioned in Example 3 are definable as long as the vocabulary of $\mathbb{A}$ contains a binary relation $<$ and the universe contains an element 5.

Our proposed solution to the requirements raised in Section 3 is to consider automata where all components (the input alphabet, the states, the transition relation, the initial and accepting states) are definable over some structure $\mathbb{A}$. Call these *definable automata* over some structure $\mathbb{A}$ [8, 11].

▶ **Running Example.** *Assume that the atoms are $\mathbb{A} = (\mathbb{N}, =)$, i.e. some countably infinite set with equality only. Here is a deterministic automaton which recognises the language from our running example, i.e. words in $\mathbb{A}^*$ with at most two distinct letters. The input alphabet is $\mathbb{A}$, which is clearly a definable set, as defined by the set-builder expression $\{a : a \in \mathbb{A}\}$. The set of states Q is*

$$\{\emptyset, \bot\} \cup \{a : a \in \mathbb{A}\} \cup \{\{a, b\} : a \neq b \in \mathbb{A}\},$$

which is also definable as long as we assume that $\bot$ is syntactic sugar for some definable set like $\{\emptyset\}$. The initial state is $\emptyset$, which is clearly definable, and all states states are final except $\bot$. Here is the set of transitions, which happens to be a total function of type $Q \times \mathbb{A} \to Q$ as required in a deterministic automaton:

$$\{(\emptyset, a, \{a\}) : a \in \mathbb{A}\} \cup$$
$$\{(\{a\}, b, \{a, b\}) : a, b \in \mathbb{A}\} \cup$$
$$\{(\{a, b\}, a, \{a, b\}) : a, b \in \mathbb{A}\} \cup$$
$$\{(\{a, b\}, c, \bot) : a, b, c \in \mathbb{A} \text{ with } a \neq b \wedge a \neq c \wedge b \neq c\} \cup$$
$$\{(\bot, a, \bot) : a \in \mathbb{A}\}$$

The above description uses ordered triples, which are formally not part of the syntax of set-builder expressions. However, triples and other tuples can encoded in sets using Kuratowski pairing:

$$(x, y) \stackrel{\text{def}}{=} \{x, \{x, y\}\} \qquad (x, y, z) \stackrel{\text{def}}{=} ((x, y), z).$$

Under the above encoding of triples as sets, it is clear that the transition relation defined above is an example of a definable set.

Based on the above example, it is easy to see that definable automata generalise register automata. What is more, the added flexibility of definable automata is sufficient to satisfy the requirements mentioned in Section 3, in particular minimisation, i.e. for every deterministic definable automaton there exists a unique (up to isomorphism of automata) minimal deterministic definable automaton which recognises the same language [8, Theorem 3.8]. (This minimisation requires an additional assumption on the atoms, called oligomorphism, which will be discussed in Section 6.)

Without further restrictions on the atoms $\mathbb{A}$, definable automata are too general to be useful, as shown in the following example.

▶ **Example 5.** Assume that the atoms are Presburger arithmetic $\mathbb{A} = (\mathbb{N}, +)$. Consider an automaton where the input alphabet has one letter only, say the input alphabet is $\{\emptyset\}$, and the set of states is the definable set of all atoms (i.e. natural numbers). Assume that there is only one final state, namely the natural number 0. Since there is only one input letter, the transition function can be viewed as a function $\mathbb{A} \to \mathbb{A}$. As the transition function, consider the function

$$a \mapsto \begin{cases} a/2 & \text{if } a \text{ is even} \\ 3a+1 & \text{otherwise} \end{cases}$$

which is a definable set as witnessed by the following set-builder expression for its graph:

$$\{(a,b) : a, b \in \mathbb{A} \text{ with } a = b + b\} \quad \cup$$
$$\{(a,b) : a, b \in \mathbb{A} \text{ with } \neg(\exists c\ a = c + c) \wedge b = a + a + a + 1\}.$$

The transition function is based on the famous Collatz conjecture, which in the terminology of this example says: for every choice of initial state, at least one input word is accepted. In fact, all decision problems, such as emptiness or equivalence, are going to be undecidable for this particular choice of atoms (Presburger arithmetic, or even $(\mathbb{N}, <)$ or $(\mathbb{N}, +1)$), which follows from the fact that Minsky machines are a special case of definable automata.

▶ **Example 6.** Assume that the atoms are arithmetic $\mathbb{A} = (\mathbb{N}, +, \times)$, and consider a set-builder expression of the form

$$\{a : a \in \mathbb{A} \text{ with } \varphi(a)\}.$$

The above set is empty if and only if $\varphi(a)$ is unsatisfiable. Since satisfiability in arithmetic is undecidable, it follows that one cannot even decide if a set-builder expression describes an empty set. (This is in contrast with Presburger arithmetic from Example 5, where at least emptiness for set-builder expressions is definable, by virtue of Presburger arithmetic having a decidable theory, see [19, Proposition 2].) Other problems, such as nonemptiness or equivalence for automata are clearly also going to be undecidable when the atoms are arithmetic.

5 Graph reachability for definable sets

In the previous section, we introduced definable sets, and discussed definable automata, i.e. automata where all components are definable sets. In Examples 5 and 6 we argued

```
input V, E, s, t given by set-builder expressions

R := {s}
repeat
 for v in R do
  for w in V do
    if {v,w} in E then
      R += {w}
until R does not grow

if t in R then
 print "reachable"
else
 print "unreachable"
```

Figure 3 A naive algorithm for reachability.

that emptiness for automata is undecidable when the atoms are $(\mathbb{N}, +)$ or $(\mathbb{N}, +, \times)$. In this section, we discuss algorithmic problems like emptiness of automata in more detail. Instead of automata emptiness, we will discuss the essentially equivalent but more fundamental problem of graph reachability. We formalise the problem and present a condition on the atom structure $\mathbb{A}$ which guarantees the graph reachability problem is decidable. This condition is going to be violated for atoms like $(\mathbb{N}, <)$, $(\mathbb{N}, +)$ and $(\mathbb{N}, +, \times)$ but it is going to be satisfied for atoms like $(\mathbb{N}, =)$ or $(\mathbb{Q}, <)$.

The graph reachability problem

For a logical structure $\mathbb{A}$, define *reachability for definable graphs over* $\mathbb{A}$ to be the following decision problem:

- **Input.** A graph (V, E) where V, E are definable sets and two vertices $s, t \in V$.
- **Question.** Is there a path from s to t?

In the decision problem above, the inputs are represented by set-builder expressions. This representation assumes that there is some way of representing the universe of $\mathbb{A}$, which is the case for all atom structures we have discussed so far, where the universe consists of natural or rational numbers. We are mainly interested in decidability and not in the precise complexity of the decision problem.

▶ **Example 7.** Assume that the atoms $\mathbb{A}$ are $(\mathbb{N}, +)$. A valid input for the reachability problem for definable graphs over $\mathbb{A}$ is

$$V = \mathbb{N} \quad E = \{(a, b) : a, b \in \mathbb{A} \text{ with } a = b + b \vee a + 1 = b\} \quad s = 7 \quad t = 2.$$

For this particular input the answer is "yes" because one can go from 7 to 2 by doing several steps of the form "divide by two or add one". For the same reason as discussed in Example 5, i.e. encoding Minsky machines, reachability is undecidable for definable graphs over Presburger arithmetic $(\mathbb{N}, +)$. Actually, the undecidability holds already for atoms $(\mathbb{N}, <)$, since Minsky machines use only increments and decrements on the counters, and these can be defined in first-order logic using only the order. Note that to get undecidability for $(\mathbb{N}, <)$ it is important that formulas in the guards of definable sets are allowed to use

quantifiers. If only quantifier-free formulas would be allowed in the guards, then graph reachability for $(\mathbb{N}, <)$ would be decidable [13].

Example 7 shows that the reachability problem is undecidable when the atoms are natural numbers with order, or any other richer structure such as addition or multiplication. What about definable graphs over atoms with equality only, or atoms such as $(\mathbb{Q}, <)$? It turns out that for these atoms, reachability is decidable, and the algorithm is quite straightforward. Define R_n to be the vertices which can be reached from the source by path of at most n edges. A naive algorithm to solve graph reachability (see Figure 3) would be to simply compute the sequence $R_0 \subseteq R_1 \subseteq R_2 \subseteq \cdots$ until it stabilises, and then test if the target vertex is in the stable set. Here is a key property of the program in Figure 3. Assuming that the atoms have decidable first-order theory (which assumption is true Presburger arithmetic $(\mathbb{N}, +)$ but not for general arithmetic $(\mathbb{N}, +, \times)$), then at least each step (both for loops) of the naive algorithm can be carried out in finite time, but the number of steps (repeat loop) might be unbounded:

need not terminate

```
repeat
 for v in R do
  for w in V do
   if {v,w} in E then
    R += {w}
until R does not grow
```

will always terminate, assuming $\mathbb{A}$ has decidable first-order theory

A more formal statement is in the following lemma.

▶ **Lemma 8.** *Let $\mathbb{A}$ be a logical structure. Suppose that $E \subseteq V \times V$ and $R \subseteq V$ are given by set-builder expressions over $\mathbb{A}$. Then one can compute a set-builder expression representing*

$$R \cup \{v \in V : \{v, w\} \in E \text{ for some } w \in R\}.$$

Assume furthermore that $\mathbb{A}$ has decidable first-order theory. Then one can decide, given R' and R represented using set-builder expressions, if $R' \subseteq R$.

The above lemma can be shown using [19, Proposition 2]; but it is mainly interesting as part of a more general framework, namely programming languages that deal with definable sets. There are currently two programming languages: a functional one [7], with an implementation as a Haskell library [20]; and an imperative one [12] with an implementation as a C++ library [19]. The example reachability program in Figure 3 is based on the imperative approach. The original version of the imperative programming language [12] assumed that the atoms were oligomorphic (see below), but the version from [19] relaxed this assumption to having a decidable first-order theory (which captures additional examples such as Presburger arithmetic). The semantics of both languages are designed so that one does not need to prove results like Lemma 8 by hand; but one can simply use more general principles like the following: every program without `repeat` can be simulated in finite time, assuming the inputs (i.e. program state before) and outputs (i.e. program state after) are represented using set-builder expressions.

As follows from Example 7, decidability of the first-order theory of $\mathbb{A}$ alone does not guarantee that the repeat loop in the program from Firgure 3 will terminate in finitely many steps. Remarkably, when the atoms have equality only, or they are $(\mathbb{Q}, <)$, then the repeat loop necessarily terminates. The reason is that atoms with equality only or $(\mathbb{Q}, <)$ are examples of *oligomorphic* structures. In the next two sections, we discuss oligomorphic structures and prove this termination (Theorem 13). The assumption that the atoms are oligomorphic is what makes the theory of definable sets robustly well behaved.

6 Oligomorphism and orbit-finiteness

In this section we describe what it means for a logical structure to be oligomorphic. The main use of this assumption is that it allows us to use relaxed notion of finiteness for sets, called *orbit-finiteness*; in particular all definable sets will turn out to be orbit-finite.

Definition of oligomorphism

If $\mathbb{A}$ is a logical structure, then an *automorphism* is defined to be any permutation of its universe which is consistent with all the relations and functions in the vocabulary of $\mathbb{A}$. For example, when $\mathbb{A}$ has only equality in its signature then an automorphism is any permutation; while if $\mathbb{A}$ is $(\mathbb{Q}, <)$ then an automorphism is any order-preserving permutation. The structures $(\mathbb{N}, <)$ and $(\mathbb{N}, +)$ have no automorphisms, while $(\mathbb{Z}, <)$ has only translations as automorphisms.

▶ **Definition 9** (Oligomorphism)**.** Consider a logical structure $\mathbb{A}$. We say that two tuples $\bar{a}, \bar{b} \in \mathbb{A}^k$ are *in the same orbit* if there exists an automorphism of $\mathbb{A}$ which takes $\bar{a}$ to $\bar{b}$ componentwise. The structure $\mathbb{A}$ is called *oligomorphic* if for every k, the "same orbit" equivalence relation on $\mathbb{A}^k$ has finitely many equivalence classes.

The notion of oligomorphism made its appearance in model theory in 1959, thanks to a theorem proved independently by Engeler, Ryll-Nardzewski and Svenonius: for a countable structure, being oligomorphic is equivalent to having an ω-categorical theory, see [16, Theorem 7.3.1]. One important corollary of this theorem (and its proof) is that in an oligomorphic structure, every orbit of $\mathbb{A}^k$ can be defined by a formula of first-order logic with k free variables; we will use this property later on.

▶ **Example 10.** Here are some examples and non-examples of oligormophic structures.

- Assume that $\mathbb{A} = (\mathbb{N}, <)$. This structure has no automorphisms, and therefore the "same orbit" equivalence relation on $\mathbb{A}$ has infinitely many equivalence classes (which are singletons). Therefore $\mathbb{A}$ is not oligomorphic.
- Assume that $\mathbb{A} = (\mathbb{Z}, <)$. The automorphisms are translations, and hence the "same orbit" equivalence relation has one equivalence class on $\mathbb{A}$. However, there are infinitely many equivalence classes for $\mathbb{A}^2$, because

  $$(a_1, a_2), (b_1, b_2) \text{ are in the same orbit} \qquad \text{iff} \qquad a_1 - a_2 = b_1 - b_2.$$

  Therefore $\mathbb{A}$ is not oligomorphic.
- Assume that $\mathbb{A} = (\mathbb{N}, =)$, i.e. a countably infinite set with equality only. For every k, tuples in $\mathbb{A}^k$ are in the same orbit if and only if they have the same equality types, and there are finitely many equality types. Therefore, $\mathbb{A}$ is oligomorphic.
- Assume that $\mathbb{A} = (\mathbb{Q}, <)$. It is not difficult to see that for every k, tuples $\bar{a}, \bar{b}$ satisfy $\bar{a} \sim \bar{b}$ if and only if they have the same order types, and there are finitely many order types. Therefore, $\mathbb{A}$ is oligomorphic.
- Every structure over a finite vocabulary without functions that is homogeneous is oligomorphic. This covers Fraïssé limits of classes of finite relational structures, such as the previous two items, or the countable random graph. For more on homogeneous structures, the Fraïssé limit and the random graph, see [16, Section 7].

Orbit-finiteness

Oligomorphic structures turn out to be exceptionally well-behaved with respect to definable sets. The reason for this is that in an oligomorphic structure one can distinguish a relaxed notion of finiteness, called *orbit-finiteness*, which is well behaved and can be used to prove results such as the termination of the algorithm Figure 3.

To define orbit-finiteness, we need to introduce a little set terminology. Fix a logical structure $\mathbb{A}$. Define the *cumulative hierarchy over* $\mathbb{A}$ to be objects that are built using atoms (i.e. elements of $\mathbb{A}$) and set brackets in a well-founded way. More precisely, for an ordinal number we define the rank α objects of the the cumulative hierarchy as follows: for $\alpha = 0$ the rank α objects are the empty set and every atom; for $\alpha > 0$ the rank α objects sets whose elements are objects of rank $< \alpha$. The cumulative hierarchy is the union of all ranks. For example, every definable set over $\mathbb{A}$ is in the cumulative hierarchy, but there are many more sets in the cumulative hierarchy (e.g. the cumulative hierarchy is closed under taking arbitrary subsets, unlike definable sets).

If π is an automorphism of $\mathbb{A}$ (actually, any function of type $\mathbb{A} \to \mathbb{A}$), then one can apply π to an object x in the cumulative hierarchy, by simply applying to the atoms that are used in x; the result is a new object $\pi(x)$ in the cumulative hierarchy with the same rank.

▶ **Definition 11** (Finite support and orbit-finiteness)**.** Let $\mathbb{A}$ be a logical structure. For x in the cumulative hierarchy over $\mathbb{A}$, we say that x is *finitely supported* if there exists a tuple of atoms $\bar{a}$ (which is called a *support* of x) such that

$$\pi(\bar{a}) = \sigma(\bar{a}) \quad \text{implies} \quad \pi(y) = \sigma(y) \qquad \text{for every automorphisms } \pi, \sigma \text{ of } \mathbb{A}.$$

We say that x is *orbit-finite* if the following equivalence relation on elements of x has finitely many equivalence classes:

$$y \sim z \quad \text{if } y = \pi(z) \qquad \text{for some automorphism } \pi \text{ of } \mathbb{A}.$$

The notion of finite supports is fundamental to set theories such as Fraenkel-Mostowski, and more recently to the theory of nominal sets [24]. To the author's best knowledge, the notion of orbit-finiteness was first explicitly proposed in [6]. In the terminology of the above definition, $\mathbb{A}$ is oligomorphic if and only if $\mathbb{A}^k$ is orbit-finite for every k. The following theorem shows that, under the assumption that the atoms are oligomorphic, then the notion of orbit-finiteness is well behaved and definable sets are the same as sets which are hereditarily orbit-finite. For a proof of the following theorem, see [5].

▶ **Theorem 12.** *Let $\mathbb{A}$ be a logical structure which is oligomorphic, and let x be in the cumulative hierarchy over $\mathbb{A}$.*

1. *x is orbit-finite if and only if for every tuple of atoms $\bar{a}$, the following equivalence relation has finitely many equivalence classes:*

$$y \sim_{\bar{a}} z \quad \text{if } y = \pi(z) \text{ for some automorphism } \pi \text{ of } \mathbb{A} \text{ which satisfies } \pi(\bar{a}) = \bar{a}.$$

2. *x is definable if and only if it is hereditarily orbit-finite (i.e. x finitely supported and orbit-finite, and both these properties also hold for elements of x, their elements, and so on recursively).*

The equivalence in item 2 of the above theorem is very useful for computation: in some cases it is more convenient to use definable sets (e.g. to represent sets in a finite way), and in some cases it is more convenient to use orbit-finite sets (e.g. in termination proofs). We now show an example of this usefulness.

Termination of the reachability algorithm

In Figure 3 from Section 5, we presented a naive reachability algorithm for definable graphs. We also remarked that, as long as the atoms have decidable first-order theory, then each iteration of the repeat loop could be evaluated in finite time. We are now ready to show that, as long as the atoms are oligomorphic, then the repeat loop will be be performed finitely often.

▶ **Theorem 13.** *Assume that $\mathbb{A}$ is oligomorphic. Then the reachability algorithm in Figure 3 terminates on every input.*

Proof. Assume that the input to is a graph with distinguished source and target, and that each part of the input (vertices, edges, source and target) is a definable set represented by a set-builder expression. By item 2 in Theorem 12, each part of the input has a finite support. By combining these finite supports into a single tuple, we can conclude there is some tuple of atoms $\bar{a}$ which supports all parts of the input, i.e. $\bar{a}$ supports the vertices, the edges and the source and target vertices. For $n \in \{0, 1, \ldots\}$ define R_n to be the vertices which are reachable from the source vertex by a path with at most n edges. Recall the equivalence relation $\sim_{\bar{a}}$ mentioned in item 1 of Theorem 12. Using induction on n and the assumption that $\bar{a}$ supports the source vertex and the edges, we get the following observation:

(*) each set R_n is a union of equivalence classes of the $\sim_{\bar{a}}$.

By item 1 of Theorem 12, there are finitely many equivalence classes. By (*) each step of the reachability algorithm adds some new equivalence classes, and therefore the algorithm must terminate in a finite number of steps. ◀

The goal of the above proof was to illustrate the interplay between definability (as a way of representing infinite inputs) and orbit-finiteness (as a way of proving termination for algorithms). Other examples of this interplay include algorithms which uses a fixpoint computation, such as the standard algorithm for emptiness of a context-free grammar (which works if the grammar is definable over oligomorphic atoms) or the Moore minimisation algorithm for deterministic automata (which works for definable automata over oligomorphic atoms).

References

1 Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

2 Michael Benedikt, Clemens Ley, and Gabriele Puppis. Automata vs. logics on data words. In *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, pages 110–124, 2010. doi:10.1007/978-3-642-15205-4_12.

3 Michael Benedikt, Clemens Ley, and Gabriele Puppis. What you must remember when processing data words. In *Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management, Buenos Aires, Argentina, May 17-20, 2010*, 2010. URL: http://ceur-ws.org/Vol-619/paper11.pdf.

4 Henrik Björklund and Thomas Schwentick. On notions of regularity for data languages. *Theor. Comput. Sci.*, 411(4-5):702–715, 2010. doi:10.1016/j.tcs.2009.10.009.

5 Mikołaj Bojańczyk. Atom book [online]. https://www.mimuw.edu.pl/~bojan/paper/atom-book.

6 Mikołaj Bojańczyk. Nominal monoids. *Theory Comput. Syst.*, 53(2):194–222, 2013. doi:10.1007/s00224-013-9464-1.

7 Mikołaj Bojańczyk, Laurent Braud, Bartek Klin, and Slawomir Lasota. Towards nominal computation. In *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 401–412, 2012. doi:10.1145/2103656.2103704.

8 Mikołaj Bojańczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014. doi:10.2168/LMCS-10(3:4)2014.

9 Mikołaj Bojańczyk, Bartek Klin, Slawomir Lasota, and Szymon Toruńczyk. Turing machines with atoms. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 183–192, 2013. doi:10.1109/LICS.2013.24.

10 Mikołaj Bojańczyk and Slawomir Lasota. A machine-independent characterization of timed languages. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 92–103, 2012. doi:10.1007/978-3-642-31585-5_12.

11 Mikołaj Bojańczyk, Luc Segoufin, and Szymon Toruńczyk. Verification of database-driven systems via amalgamation. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA – June 22-27, 2013*, pages 63–74, 2013. doi:10.1145/2463664.2465228.

12 Mikołaj Bojańczyk and Szymon Toruńczyk. Imperative programming in sets with atoms. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 4–15, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.4.

13 Karlis Cerans. Deciding properties of integral relational automata. In *Automata, Languages and Programming, 21st International Colloquium, ICALP94, Jerusalem, Israel, July 11-14, 1994, Proceedings*, pages 35–46, 1994. doi:10.1007/3-540-58201-0_56.

14 Lorenzo Clemente and Slawomir Lasota. Reachability analysis of first-order definable pushdown systems. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, pages 244–259, 2015. doi:10.4230/LIPIcs.CSL.2015.244.

15 Lorenzo Clemente and Slawomir Lasota. Timed pushdown automata revisited. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 738–749, 2015. doi:10.1109/LICS.2015.73.

16 W. Hodges. *Model Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1993.

17 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

18 Bartek Klin, Slawomir Lasota, Joanna Ochremiak, and Szymon Toruńczyk. Turing machines with atoms, constraint satisfaction problems, and descriptive complexity. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14-18, 2014*, pages 58:1–58:10, 2014. doi:10.1145/2603088.2603135.

19 Eryk Kopczynski and Szymon Toruńczyk. LOIS: syntax and semantics. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 586–598, 2017. URL: http://dl.acm.org/citation.cfm?id=3009876.

20 Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michal Szynwelski. Learning nominal automata. In *Proceedings of the 44th ACM SIGPLAN Symposium on*

Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017, pages 613–625, 2017. URL: http://dl.acm.org/citation.cfm?id=3009879.

21 Andrzej S. Murawski, Steven J. Ramsay, and Nikos Tzevelekos. Bisimilarity in fresh-register automata. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 156–167, 2015. doi:10.1109/LICS.2015.24.

22 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.

23 M. Pistore. *History Dependent Automata*. PhD thesis, University of Pisa, 1999.

24 A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2013.

25 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, pages 41–57, 2006. doi:10.1007/11874683_3.

26 Victor Vianu. Automatic verification of database-driven systems: a new frontier. In *Intl. Conf. on Database Theory (ICDT)*, pages 1–13, 2009. doi:10.1145/1514894.1514896.

27 Jin yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992. doi:10.1007/BF01305232.

Efficient Algorithms for Graph-Related Problems in Computer-Aided Verification

Monika Henzinger

Faculty of Computer Science, University of Vienna, Vienna, Austria
monika.henzinger@univie.ac.at

Abstract

Fundamental algorithmic problems that lie in the core of many application in formal verification and analysis of systems can be described as graph-related algorithmic problems. Nodes in these problems are of one of two (or three) types, giving rise to a game-theoretic viewpoint: *Player one* nodes are under the control of the algorithm that wants to accomplish a goal, *player two* nodes are under the control of a worst-case adversary that tries to keep player one to achieve her goal, and *random* nodes are under the control of a random process that is oblivious to the goal of player one. A graph containing only player one and random nodes is called a Markov Decision Process, a graph containing only player one and player two nodes is called a game graph. A variety of goals on these graphs are of interest, the simplest being whether a fixed set of nodes can be reached. The algorithmic question is then whether there is a strategy for player one to achieve her goal from a given starting node. In this talk we give an overview of a variety of goals that are interesting in computer-aided verification and present upper and (conditional) lower bounds on the time complexity for deciding whether a winning strategy for player one exists.

1998 ACM Subject Classification F.1.1 Models of Computation, D.2.4 Software/Program Verification, I.1.2 Algorithms

Keywords and phrases Computer-aided Verification, Game Theory, Markov Decision Process

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.2

Category Invited Talk

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 2; pp. 2:1–2:1

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

Local Computation Algorithms

Ronitt Rubinfeld

MIT, Cambridge, MA, USA; and
Tel Aviv University, Tel Aviv, Israel
ronitt@csail.mit.edu

Abstract

Consider a setting in which inputs to and outputs from a computational problem are so large, that there is not time to read them in their entirety. However, if one is only interested in small parts of the output at any given time, is it really necessary to solve the entire computational problem? Is it even necessary to view the whole input? We survey recent work in the model of *local computation algorithms* which for a given input, supports queries by a user to values of specified bits of a legal output. The goal is to design local computation algorithms in such a way that very little of the input needs to be seen in order to determine the value of any single bit of the output. In this talk, we describe results on a variety of problems for which sublinear time and space local computation algorithms have been developed – we will give special focus to finding maximal independent sets and sparse spanning graphs.

1998 ACM Subject Classification E.1 Data Structures, F.2 Analysis of Algorithms and Problem Complexity, I.1.2 Algorithms

Keywords and phrases Massive Data Sets, Approximate Solutions, Maximal Independent Set, Sparse Spanning Graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.3

Category Invited Talk

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 3; pp. 3:1–3:1

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

Fast and Powerful Hashing Using Tabulation*†

Mikkel Thorup

Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
mikkel2thorup@gmail.com

Abstract

Randomized algorithms are often enjoyed for their simplicity, but the hash functions employed to yield the desired probabilistic guarantees are often too complicated to be practical. Here we survey recent results on how simple hashing schemes based on tabulation provide unexpectedly strong guarantees.

Simple tabulation hashing dates back to Zobrist [1970]. Keys are viewed as consisting of c characters and we have precomputed character tables $h_1, ..., h_q$ mapping characters to random hash values. A key $x = (x_1, ..., x_c)$ is hashed to $h_1[x_1] \oplus h_2[x_2] ..... \oplus h_c[x_c]$. This schemes is very fast with character tables in cache. While simple tabulation is not even 4-independent, it does provide many of the guarantees that are normally obtained via higher independence, e.g., linear probing and Cuckoo hashing.

Next we consider *twisted tabulation* where one character is "twisted" with some simple operations. The resulting hash function has powerful distributional properties: Chernoff-Hoeffding type tail bounds and a very small bias for min-wise hashing.

Finally, we consider *double tabulation* where we compose two simple tabulation functions, applying one to the output of the other, and show that this yields very high independence in the classic framework of Carter and Wegman [1977]. In fact, w.h.p., for a given set of size proportional to that of the space consumed, double tabulation gives fully-random hashing.

While these tabulation schemes are all easy to implement and use, their analysis is not.

This keynote talk surveys results from the papers in the reference list.

1998 ACM Subject Classification E.1 [Data Structures] Tables, E.2 [Data Storage Representations] Hash Table Representations, F.2.2 [Analysis of Algorithms and Problem Complexity] Nonnumerical Algorithms and Problems – Sorting and Searching, H.3 [Information Storage and Retrieval] Information Search and Retrieval – Search Process

Keywords and phrases Hashing, Randomized Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.4

Category Invited Talk

* Research partly supported by Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research.

† A similar talk abstract also appears in *Proceedings of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 1:1–1:2, 2016 – http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2016.1 –, and in *Proceeding of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA), page 1–1, 2017.*

References

1. Tobias Christiani, Rasmus Pagh, and Mikkel Thorup. From independence to expansion and back again. In *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*, pages 813–820, 2015.
2. Søren Dahlgaard, Mathias Bæk Tejs Knudsen, Eva Rotenberg, and Mikkel Thorup. The power of two choices with simple tabulation. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1631–1642, 2016.
3. Søren Dahlgaard and Mikkel Thorup. Approximately minwise independence with twisted tabulation. In *Proc. 14th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 134–145, 2014.
4. Søren Dahlgaard, Mathias Bæk Tejs Knudsen, Eva Rotenberg, and Mikkel Thorup. Hashing for statistics over k-partitions. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1292–1310, 2015.
5. Mihai Pătraşcu and Mikkel Thorup. The power of simple tabulation-based hashing. *Journal of the ACM*, 59(3):Article 14, 2012. Announced at STOC'11.
6. Mihai Pătraşcu and Mikkel Thorup. Twisted tabulation hashing. In *Proc. 24th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 209–228, 2013.
7. Mikkel Thorup. Simple tabulation, fast expanders, double tabulation, and high independence. In *Proc. 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 90–99, 2013.

Optimal Unateness Testers for Real-Valued Functions: Adaptivity Helps*†

Roksana Baleshzar[1], Deeparnab Chakrabarty[2], Ramesh Krishnan S. Pallavoor[3], Sofya Raskhodnikova[4], and C. Seshadhri[5]

1 **Department of Computer Science and Engineering, Pennsylvania State University, State College, PA, USA**
rxb5410@cse.psu.edu
2 **Department of Computer Science, Dartmouth College, Hanover, NH, USA**
deeparnab@dartmouth.edu
3 **Department of Computer Science and Engineering, Pennsylvania State University, State College, PA, USA**
ramesh@psu.edu
4 **Department of Computer Science and Engineering, Pennsylvania State University, State College, PA, USA**
sofya@cse.psu.edu
5 **Department of Computer Science, University of California – Santa Cruz, Santa Cruz, CA, USA**
sesh@ucsc.edu

Abstract

We study the problem of testing unateness of functions $f : \{0,1\}^d \to \mathbb{R}$. We give an $O(\frac{d}{\varepsilon} \cdot \log \frac{d}{\varepsilon})$-query nonadaptive tester and an $O(\frac{d}{\varepsilon})$-query adaptive tester and show that both testers are optimal for a fixed distance parameter ε. Previously known unateness testers worked only for Boolean functions, and their query complexity had worse dependence on the dimension both for the adaptive and the nonadaptive case. Moreover, no lower bounds for testing unateness were known. We generalize our results to obtain optimal unateness testers for functions $f : [n]^d \to \mathbb{R}$.

Our results establish that adaptivity helps with testing unateness of real-valued functions on domains of the form $\{0,1\}^d$ and, more generally, $[n]^d$. This stands in contrast to the situation for monotonicity testing where there is no adaptivity gap for functions $f : [n]^d \to \mathbb{R}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Property testing, unate and monotone functions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.5

1 Introduction

We study the problem of testing whether a given real-valued function f on domain $[n]^d$, where $n, d \in \mathbb{N}$, is unate. A function $f : [n]^d \to \mathbb{R}$ is *unate* if for every coordinate $i \in [d]$, the function is either nonincreasing in the coordinate i or nondecreasing in the coordinate i. Unate functions naturally generalize monotone functions, which are nondecreasing in all

* A preliminary full version of this paper appears in ECCC [2], https://eccc.weizmann.ac.il/report/2017/049/.

† The authors R.B., R.P. and S.R. were partially supported by NSF award CCF-1422975. This work was done while D.C. was at Microsoft Research, India.

coordinates, and **b**-monotone functions, which have a particular direction in each coordinate (either nonincreasing or nondecreasing), specified by a bit-vector $\mathbf{b} \in \{0,1\}^d$. More precisely, a function is **b**-monotone if it is nondecreasing in coordinates i with $\mathbf{b}_i = 0$ and nonincreasing in the other coordinates. Observe that a function f is unate iff there exists some $\mathbf{b} \in \{0,1\}^d$ for which f is **b**-monotone.

A *tester* [33, 25] for a property $\mathcal{P}$ of a function f is an algorithm that gets a distance parameter $\varepsilon \in (0,1)$ and query access to f. It has to accept with probability at least $2/3$ if f has property $\mathcal{P}$ and reject with probability at least $2/3$ if f is ε-far (in Hamming distance) from $\mathcal{P}$. We say that f is ε-far from $\mathcal{P}$ if at least an ε fraction of values of f must be modified to make f satisfy $\mathcal{P}$. A tester has *one-sided error* if it always accepts a function satisfying $\mathcal{P}$, and has *two-sided error* otherwise. A *nonadaptive* tester makes all its queries at once, while an *adaptive* tester can make queries after seeing answers to the previous ones.

Testing of various properties of functions, including monotonicity (see, e.g., [24, 19, 20, 31, 22, 21, 26, 1, 27, 3, 8, 7, 10, 13, 9, 6, 14, 15, 12, 17, 16, 29, 4, 5, 18] and recent surveys [32, 11]), the Lipschitz property [28, 13, 9, 16], bounded-derivative properties [12], and unateness [24, 30], has been studied extensively over the past two decades. Even though unateness testing was initially discussed in the seminal paper by Goldreich et al. [24] that gave first testers for properties of functions, relatively little is known about testing this property. All previous work on unateness testing focused on the special case of Boolean functions on domain $\{0,1\}^d$. The domain $\{0,1\}^d$ is called the *hypercube* and the more general domain $[n]^d$ is called the *hypergrid*. Goldreich et al. [24] provided a $O(\frac{d^{3/2}}{\varepsilon})$-query nonadaptive tester for unateness of Boolean functions on the hypercube. Recently, Khot and Shinkar [30] improved the query complexity to $O(\frac{d \log d}{\varepsilon})$, albeit with an adaptive tester.

In this paper, we improve upon both these works, and our results hold for a more general class of functions. Specifically, we show that unateness of real-valued functions on hypercubes can be tested nonadaptively with $O(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon})$ queries and adaptively with $O(\frac{d}{\varepsilon})$ queries. More generally, we describe a $O(\frac{d}{\varepsilon} \cdot (\log \frac{d}{\varepsilon} + \log n))$-query nonadaptive tester and a $O(\frac{d \log n}{\varepsilon})$-query adaptive tester of unateness of real-valued functions over hypergrids.

In contrast to the state of knowledge for unateness testing, the complexity of testing monotonicity of real-valued functions over the hypercube and the hypergrid has been resolved. For constant distance parameter ε, it is known to be $\Theta(d \log n)$. Moreover, this bound holds for all *bounded-derivative* properties [12], a large class that includes **b**-monotonicity and some properties quite different from monotonicity, such as the Lipschitz property. Amazingly, the upper bound for all these properties is achieved by the same simple and, in particular, nonadaptive, tester. Even though proving lower bounds for adaptive testers has been challenging in general, a line of work, starting from Fischer [21] and including [8, 14, 12], has established that adaptivity does not help for this large class of properties. Since unateness is so closely related, it is natural to ask whether the same is true for testing unateness.

We answer this in the negative: we prove that any non-adaptive tester of real valued functions over the hypercube (for some constant distance parameter) must make $\Omega(d \log d)$ queries. More generally, it needs $\Omega(d(\log d + \log n))$ queries for the hypergrid domain. These lower bounds complement our algorithms, completing the picture for unateness testing of real-valued functions. From a property testing standpoint, our results establish that unateness is different from monotonicity and, more generally, any derivative-bounded property.

1.1 Formal Statements and Technical Overview

Our upper bounds are summarized in the following theorem, stated for functions over the hypergrid domains. (Recall that the hypercube is a special case of the hypergrid with $n = 2$.)

▶ **Theorem 1.1.** *Consider functions $f : [n]^d \to \mathbb{R}$ and a distance parameter $\varepsilon \in (0, 1/2)$.*
1. *There is a nonadaptive unateness tester that makes $O(\frac{d}{\varepsilon}(\log \frac{d}{\varepsilon} + \log n))$ queries*[1].
2. *There is an adaptive unateness tester that makes $O(\frac{d \log n}{\varepsilon})$ queries.*

Both testers have one-sided error.

Our main technical contribution is the proof that the extra $\Omega(\log d)$ is needed for nonadaptive testers. This result demonstrates a gap between adaptive and nonadaptive unateness testing.

▶ **Theorem 1.2.** *Any nonadaptive unateness tester (even with two-sided error) for real-valued functions $f : \{0,1\}^d \to \mathbb{R}$ with distance parameter $\varepsilon = 1/8$ must make $\Omega(d \log d)$ queries.*

The lower bound for adaptive testers is an easy adaptation of the monotonicity lower bound in [14]. We state this theorem for completeness and prove it in the full version [2].

▶ **Theorem 1.3.** *Any unateness tester for functions $f : [n]^d \to \mathbb{R}$ with distance parameter $\varepsilon \in (0, 1/4)$ must make $\Omega\left(\frac{d \log n}{\varepsilon} - \frac{\log 1/\varepsilon}{\varepsilon}\right)$ queries.*

Theorems 1.2 and 1.3 directly imply that our nonadaptive tester is optimal for constant ε, even for the hypergrid domain. All missing proofs and details from the technical sections appear in the full version of this paper [2].

1.1.1 Overview of Techniques

We first consider the hypercube domain. For each $i \in [d]$, an *i-edge* of the hypercube is a pair (x, y) of points in $\{0,1\}^d$, where $x_i = 0, y_i = 1$, and $x_j = y_j$ for all $j \in ([d] \setminus \{i\})$. Given an input function $f : \{0,1\}^d \to \mathbb{R}$, we say an i-edge (x, y) is *increasing* if $f(x) < f(y)$, *decreasing* if $f(x) > f(y)$, and *constant* if $f(x) = f(y)$.

Our nonadaptive unateness tester on the hypercube uses the work investment strategy from [6] (also refer to Section 8.2.4 of Goldreich's book [23]) to "guess" a good dimension where to look for violations of unateness (specifically, both increasing and decreasing edges). For all $i \in [d]$, let α_i be the fraction of the i-edges that are decreasing, β_i be the fraction of the i-edges that are increasing, and $\mu_i = \min(\alpha_i, \beta_i)$. The dimension reduction theorem from [12] implies that if the input function is ε-far from unate, then the average of μ_i over all dimensions is at least $\frac{\varepsilon}{4d}$. If the tester knew which dimension had $\mu_i = \Omega(\varepsilon/d)$, it could detect a violation with high probability by querying the endpoints of $O(1/\mu_i) = O(d/\varepsilon)$ uniformly random edges. However, the tester does not know which μ_i is large and, intuitively, nonadaptively checks the following $\log d$ different scenarios, one for each $k \in [\log d]$: exactly 2^k different μ_i's are $\varepsilon/2^k$, and all others are 0. This leads to the query complexity of $O(\frac{d \log d}{\varepsilon})$.

With adaptivity, this search through $\log d$ different scenarios is not required. A pair of queries in each dimension detects influential coordinates (i.e., dimensions with many non-constant edges), and the algorithm focuses on finding violations among those coordinates. This leads to the query complexity of $O(d/\varepsilon)$, removing the $\log d$ factor.

It is relatively easy to extend (both adaptive and nonadaptive) testers from hypercubes to hypergrids by incurring an extra factor of $\log n$ in the query complexity. The role of i-edges is now played by *i-lines*. An i-line is a set of n domain points that differ only on coordinate i. The domain $[n]$ is called a line. Monotonicity on the line (a.k.a. sortedness) can be tested with $O(\frac{\log n}{\varepsilon})$ queries, using, for example, the classical *tree tester* from [20].

[1] For many properties, when the domain is extended from the hypercube to the hypergrid, testers incur an extra multiplicative factor of $\log n$ in the query complexity. This is the case for our adaptive tester. However, note that the complexity of nonadaptive unateness testing (for constant ε) is $\Theta(d(\log d + \log n))$ rather than $\Theta(d \log d \log n)$.

Instead of sampling a random i-edge, we sample a random i-line ℓ and run the tree tester on the restriction $f_{|\ell}$ of function f to the line ℓ. This is optimal for adaptive testers, but, interestingly, not for nonadaptive testers. We show that for each function f on the line that is ε-far from unateness, one of the two scenarios happen: (1) the tree tester is likely to find a violation of unateness; (2) function f is increasing (and also decreasing) on a constant fraction of pairs in $[n]$. This new angle on the classical tester allows us to replace the factor $(\log d)(\log n)$ with $\log d + \log n$ in the query complexity. Thus, the nonadaptive complexity becomes $O(d(\log d + \log n))$, which we show is optimal.

The nonadaptive lower bound. Our most significant finding is the $\log d$ gap in the query complexity between adaptive and nonadaptive testing of unateness. By previous work [21, 14], it suffices to prove lower bounds for *comparison-based* testers, i.e., testers that can only perform comparisons of the function values at queried points, but cannot use the values themselves. Our main technical contribution is the $\Omega(d \log d)$ lower bound for nonadaptive comparison-based testers of unateness on hypercube domains.

Intuitively, we wish to construct $K = \Theta(\log d)$ families of functions where, for each $k \in [K]$, functions in the k^{th} family have 2^k dimensions i with $\mu_i = \Theta(1/2^k)$, while $\mu_i = 0$ for all other dimensions. What makes the construction challenging is the existence of a *single, universal* nonadaptive $O(d)$-tester for all $\mathbf{b}$-monotonicity properties, proven in [12]. In other words, there is a single distribution on $O(d)$ queries that defines a nonadaptive property tester for $\mathbf{b}$-monotonicity, regardless of $\mathbf{b}$. Since unateness is the union of all $\mathbf{b}$-monotonicity properties, our construction must be able to fool such algorithms. Furthermore, nonadaptivity must be critical, since we obtained a $O(d)$-query adaptive tester for unateness.

Another obstacle is that once a tester finds a non-constant edge in each dimension, the problem reduces to testing $\mathbf{b}$-monotonicity for a vector $\mathbf{b}$ determined by the directions (increasing or decreasing) of the non-constant edges. That is, intuitively, most edges in our construction must be constant. This is one of the main technical challenges. The previous lower bound constructions for monotonicity testing [8, 14] crucially used the fact that all edges in the hard functions were non-constant.

We briefly describe how we overcome the problems mentioned above. By Yao's minimax principle, it suffices to construct **Yes** and **No** distributions that a deterministic nonadaptive tester cannot distinguish. First, for some parameter m, we partition the hypercube into m subcubes based of the first $\log_2 m$ most significant coordinates. Both distributions, **Yes** and **No**, sample a uniform k from $[K]$, where $K = \Theta(\log d)$, and a set $R \subseteq [d]$ of cardinality 2^k. Furthermore, each subcube $j \in [m]$ selects an "action dimension" $r_j \in R$ uniformly at random. For both distributions, in any particular subcube j, the function value is completely determined by the coordinates *not in* R, and the random coordinate $r_j \in R$. Note that all the i-edges for $i \in (R \setminus \{r_j\})$ are constant. Within the subcube, the function is a linear function with exponentially increasing coefficients. In the **Yes** distribution, any two cubes j, j' with the same action dimension orient the edges in that dimension the same way (both increasing or both decreasing), while in the **No** distribution each cube decides on the orientation independently. The former correlation maintains unateness while the latter independence creates distance to unateness. We prove that to distinguish the distributions, any comparison-based nonadaptive tester must find two distinct subcubes with the same action dimension r_j and, furthermore, make a specific query (in both) that reveals the coefficient of r_j. We show that, with $o(d \log d)$ queries, the probability of this event is negligible.

Algorithm 1: The Nonadaptive Unateness Tester over Hypercubes

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : \{0,1\}^d \to \mathbb{R}$.

for $r = 1$ *to* $\lceil 3 \log(4d/\varepsilon) \rceil$ **do**
  repeat $s_r = \lceil \frac{16d \ln 4}{\varepsilon \cdot 2^r} \rceil$ **times**
    Sample a dimension $i \in [d]$ uniformly at random.
    Sample $3 \cdot 2^r$ i-edges uniformly and independently at random and **reject** if there exists an increasing edge and a decreasing edge among the sampled edges.
accept

2 Upper Bounds

In this section, we prove parts 1–2 of Theorem 1.1, starting from the hypercube domain.

Recall the definition of i-edges and i-lines from Section 1.1.1 and what it means for an edge to be increasing, decreasing, and constant.

The starting point for our algorithms is the dimension reduction theorem from [12]. It bounds the distance of $f : [n]^d \to \mathbb{R}$ to monotonicity in terms of average distances of restrictions of f to one-dimensional functions.

▶ **Theorem 2.1** (Dimension Reduction, Theorem 1.8 in [12]). *Fix a bit vector $\mathbf{b} \in \{0,1\}^d$ and a function $f : [n]^d \to \mathbb{R}$ which is ε-far from $\mathbf{b}$-monotonicity. For all $i \in [d]$, let μ_i be the average distance of $f_{|\ell}$ to $\mathbf{b}_i$-monotonicity over all i-lines ℓ. Then $\sum_{i=1}^{d} \mu_i \geq \varepsilon/4$.*

For the special case of the hypercube domains, i-lines become i-edges, and the average distance μ_i to $\mathbf{b}_i$-monotonicity is the fraction of i-edges on which the function is not $\mathbf{b}_i$-monotone.

2.1 The Nonadaptive Tester over the Hypercube

We now describe Algorithm 1, the nonadaptive tester for unateness over the hypercubes.

It is evident that Algorithm 1 is a nonadaptive, one-sided error tester. Furthermore, its query complexity is $O\left(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon}\right)$. It suffices to prove the following.

▶ **Lemma 2.2.** *If f is ε-far from unate, Algorithm 1 rejects with probability at least $2/3$.*

Proof. Recall that α_i is the fraction of i-edges that are decreasing, β_i is the fraction of i-edges that are increasing and $\mu_i = \min(\alpha_i, \beta_i)$.

Define the d-dimensional bit vector $\mathbf{b}$ as follows: for each $i \in [d]$, let $\mathbf{b}_i = 0$ if $\alpha_i < \beta_i$ and $\mathbf{b}_i = 1$ otherwise. Observe that the average distance of f to $\mathbf{b}_i$-monotonicity over a random i-edge is precisely μ_i. Since f is ε-far from being unate, f is also ε-far from being $\mathbf{b}$-monotone. By Theorem 2.1, $\sum_{i \in [d]} \mu_i \geq \frac{\varepsilon}{4}$. Hence, $\mathbf{E}_{i \in [d]}[\mu_i] \geq \frac{\varepsilon}{4d}$. We now apply the work investment strategy due to Berman et al. [6] to get an upper bound on the probability that Algorithm 1 fails to reject.

▶ **Theorem 2.3** ([6]). *For a random variable $X \in [0,1]$ with $\mathbf{E}[X] \geq \mu$ for $\mu < \frac{1}{2}$, let $p_r = \Pr[X \geq 2^{-r}]$ and $\delta \in (0,1)$ be the desired error probability. Let $s_r = \frac{4 \ln 1/\delta}{\mu \cdot 2^r}$. Then,*

$$\prod_{r=1}^{\lceil 3 \log(1/\mu) \rceil} (1 - p_r)^{s_r} \leq \delta.$$

Algorithm 2: The Adaptive Unateness Tester over Hypercubes

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : \{0,1\}^d \to \mathbb{R}$.

1 **repeat** $10/\varepsilon$ **times**
2 **for** $i = 1$ *to* d **do**
3 Sample an i-edge e_i uniformly at random.
4 **if** e_i *is non-constant (i.e., increasing or decreasing)* **then**
5 Sample i-edges uniformly at random till we obtain a non-constant edge e_i'.
6 **reject** if one of the edges e_i, e_i' is increasing and the other is decreasing.
7 **accept**

Consider running Algorithm 1 on a function f that is ε-far from unate. Let $X = \mu_i$ where i is sampled uniformly at random from $[d]$. Then $\mathbf{E}[X] \geq \frac{\varepsilon}{4d}$. Applying the work investment strategy (Theorem 2.3) on X with $\mu = \frac{\varepsilon}{4d}$, we get that the probability that, in some iteration, Step 3 samples a dimension i such that $\mu_i \geq 2^{-r}$ is at least $1-\delta$. We set $\delta = 1/4$. Conditioned on sampling such a dimension, the probability that Step 4 fails to obtain an increasing edge and a decreasing edge among its $3 \cdot 2^r$ samples is at most $2\left(1 - 1/2^r\right)^{3 \cdot 2^r} \leq 2e^{-3} < 1/9$, as the fraction of both increasing and decreasing edges is at least $1/2^r$. Hence, the probability that Algorithm 1 rejects f is at least $\frac{3}{4} \cdot \frac{8}{9} = \frac{2}{3}$. This completes the proof of Lemma 2.2. ◀

2.2 The Adaptive Tester over the Hypercube

We now describe Algorithm 2, an adaptive tester for unateness over the hypercube domain with good expected query complexity. The final tester is obtained by repeating this tester and accepting if the number of queries exceeds a specified bound.

▶ **Claim 2.4.** *The expected number of queries made by Algorithm 2 is* $40d/\varepsilon$.

Proof. Consider one iteration of the **repeat**-loop in Step 1. We prove that the expected number of queries in this iteration is $4d$. The number of queries in Step 3 is $2d$, as 2 points per dimension are queried. Let E_i be the event that edge e_i is non-constant and T_i be the random variable for the number of i-edges sampled in Step 5. Then $\mathbf{E}[T_i] = \frac{1}{\alpha_i + \beta_i} = \frac{1}{\Pr[E_i]}$. Therefore, the expected number of all edges sampled in Step 5 is $\sum_{i=1}^{d} \Pr[E_i] \cdot \mathbf{E}[T_i] = \sum_{i=1}^{d} \Pr[E_i] \cdot \frac{1}{\Pr[E_i]} = d$. Hence, the expected number of queries in Step 5 is $2d$. Since there are $10/\varepsilon$ iterations in Step 1, the expected number of queries in Algorithm 2 is $40d/\varepsilon$. ◀

▶ **Claim 2.5.** *If f is ε-far from unate, Algorithm 2 accepts with probability at most* $1/6$.

Proof. First, we bound the probability that a violation of unateness is detected in some dimension $i \in [d]$ in one iteration of the **repeat**-loop. Consider the probability of finding a decreasing i-edge in Step 3, and an increasing i-edge in Step 5. The former is exactly α_i, and the latter is $\beta_i/(\alpha_i + \beta_i)$. Therefore, the probability we detect a violation from dimension i is $2\alpha_i\beta_i/(\alpha_i + \beta_i) \geq \min(\alpha_i, \beta_i) = \mu_i$. The probability that we fail to detect a violation in any of the d dimensions is at most $\prod_{i=1}^{d}(1 - \mu_i) \leq \exp\left(-\sum_{i=1}^{d} \mu_i\right)$, which is at most $e^{-\varepsilon/4}$ by Theorem 2.1 (Dimension Reduction). By Taylor expansion of $e^{-\varepsilon/4}$, the probability of finding a violation in one iteration is at least $1 - e^{-\varepsilon/4} \geq \frac{\varepsilon}{4} - \frac{\varepsilon^2}{32} \geq \frac{\varepsilon}{5}$. The probability that Algorithm 2 does not reject in any iteration is at most $(1 - \varepsilon/5)^{10/\varepsilon} < 1/6$. ◀

Proof of Theorem 1.1, part 2 (the special case of the hypercube domain). We run Algorithm 2, aborting and accepting if we ever make more than $240d/\varepsilon$ queries. By Markov's

Algorithm 3: Tree Tester

input : Query access to a function $h : [n] \mapsto \mathbb{R}$.

Pick $x \in [n]$ uniformly at random.
Let $Q_x \subseteq [n]$ be the set of points visited in a binary search for x. Query h on all points in Q_x.
If there is an increasing pair in Q_x, set $\mathsf{dir} \leftarrow \{\uparrow\}$; otherwise, $\mathsf{dir} \leftarrow \emptyset$.
If there is a decreasing pair in Q_x, update $\mathsf{dir} \leftarrow \mathsf{dir} \cup \{\downarrow\}$.
Return dir.

Algorithm 4: The Adaptive Unateness Tester over Hypergrids

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : [n]^d \to \mathbb{R}$.

repeat $10/\varepsilon$ **times**
  for $i = 1$ *to* d **do**
    Sample an i-line ℓ_i uniformly at random.
    Let dir_i be the output of Algorithm 3 on $f_{|\ell_i}$.
    if $\mathsf{dir}_i \neq \emptyset$ **then**
      Sample i-lines uniformly at random and run Algorithm 3 on f restricted to each line until it returns a non-empty set. Call it dir_i'.
      If $\mathsf{dir}_i \cup \mathsf{dir}_i' = \{\uparrow, \downarrow\}$, **reject**.
accept

inequality, the probability of aborting is at most 1/6. By Claim 2.5, if f is ε-far from unate, Algorithm 2 accepts with probability at most 1/6. The theorem follows by a union bound. ◀

2.3 Extension to Hypergrids

We start by establishing terminology for lines and pairs. Consider a function $f : [n]^d \to \mathbb{R}$. Recall the definition of i-lines from Section 1.1.1. A pair of points that differ only in coordinate i is called an i-pair. An i-pair (x, y) with $x_i < y_i$ is called *increasing* if $f(x) < f(y)$, *decreasing* if $f(x) > f(y)$, and *constant* if $f(x) = f(y)$.

The main tool for extending Algorithms 1 and 2 to work on hypergrids is the *tree tester*, designed by Ergun et al. [20] to test monotonicity of functions $h : [n] \to \mathbb{R}$. We modify the tree tester to return information about directions it observed instead of just accepting or rejecting. See Algorithm 3. We call a function $h : [n] \to \mathbb{R}$ *antimonotone* if $f(x) \geq f(y)$ for all $x < y$. The following lemma summarizes the guarantee of the tree tester.

▶ **Lemma 2.6** ([20, 12]). *If $h : [n] \mapsto \mathbb{R}$ is ε-far from monotone (respectively, antimonotone), then the output of Algorithm 3 on h contains $\downarrow$ (respectively, $\uparrow$) with probability at least ε.*

Our hypergrid testers are stated in Algorithms 4 and 5. Next, we explain how Lemma 2.6 and Theorem 2.1 are used in the analysis of the adaptive tester. For a dimension $i \in [d]$, let α_i and β_i denote the average distance of $f_{|\ell}$ to monotonicity and antimonotonicity, respectively, over all i-lines ℓ. Then $\mu_i := \min(\alpha_i, \beta_i)$ is the average fraction of points per i-line that needs to change to make f unate. Define the $\mathbf{b}$-vector with $\mathbf{b}_i = 0$ if $\alpha_i < \beta_i$, and $\mathbf{b}_i = 1$ otherwise. By Theorem 2.1, if f is ε-far from unate, and thus ε-far from $\mathbf{b}$-monotone, then $\sum_{i=1}^{d} \mu_i \geq \varepsilon/4$. By Lemma 2.6, the probability that the output of Algorithm 3 on $f_{|\ell}$ contains

Algorithm 5: The Nonadaptive Unateness Tester over Hypergrids

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : [n]^d \to \mathbb{R}$.

repeat $220/\varepsilon$ **times**
  for $i = 1$ *to* d **do**
    Sample an i-line ℓ uniformly at random.
    Reject if Algorithm 3, on input $f_{|\ell}$, returns $\{\uparrow, \downarrow\}$.
for $r = 1$ *to* $\lceil 3 \log(200d/\varepsilon) \rceil$ **do**
  repeat $s_r = \lceil \frac{800d \ln 4}{\varepsilon \cdot 2^r} \rceil$ **times**
    Sample a dimension $i \in [d]$ uniformly at random.
    Sample $3 \cdot 2^r$ i-pairs uniformly and independently at random.
    If we find an increasing and a decreasing pair among the sampled pairs, **reject**.
accept

$\downarrow$ (respectively, $\uparrow$), where ℓ is a uniformly random i-line, is at least α_i (respectively, β_i). The rest of the analysis of Algorithm 4 is similar to that in the hypercube case.

To analyze the nonadaptive tester, we prove Lemma 2.7, which demonstrates the power of the tree tester and may be of independent interest.

▶ **Lemma 2.7.** *Consider a function $h : [n] \to \mathbb{R}$ which is ε-far from monotone (respectively, antimonotone). At least one of the following holds:*

1. $\Pr[$*Algorithm 3, on input h, returns* $\{\uparrow, \downarrow\}] \geq \varepsilon/25$.
2. $\Pr_{u,v \in [n]}[(u, v)$ *is a decreasing (respectively, increasing) pair*$] \geq \varepsilon/25$.

3 The Lower Bound for Nonadaptive Testers over Hypercubes

In this section, we prove Theorem 1.2, which gives a lower bound for nonadaptive unateness testers for functions over the hypercube.

Previous work of [14] on lower bounds for monotonicity testing shows that, for a special class of properties, which includes unateness, it is sufficient to prove lower bounds for *comparison-based testers*. Comparison-based testers base their decisions only on the *order* of the function values at queried points, and not on the values themselves.

We first state the reduction to comparison-based testers from [14]. Let a (t, ε, δ)-tester for a property $\mathcal{P}$ be a 2-sided error t-query tester, with distance parameter ε, that errs with probability at most δ. Consider functions of the form $f : D \to \mathbb{R}$, where D is an arbitrary partial order (in particular the hypergrid/hypercube). A property $\mathcal{P}$ is invariant under monotone transformations if, for all strictly increasing maps $\phi : \mathbb{R} \to \mathbb{R}$ and all functions f, it holds that $\mathsf{dist}(f, \mathcal{P}) = \mathsf{dist}(\phi \circ f, \mathcal{P})$. In particular, unateness is invariant under monotone transformations. The following theorem is implicitly proven in [14]. Specifically, Theorem 2.1 of [14] is stated for monotonicity, but the proof only uses the fact that monotonicity is a property invariant under monotone transformations, so it applies to all such properties.

▶ **Theorem 3.1** (implicit in [21, 14])**.** *Let $\mathcal{P}$ be a property invariant under monotone transformations. Suppose there exists a nonadaptive (resp., adaptive) (t, ε, δ)-tester for $\mathcal{P}$. Then there exists a nonadaptive (resp., adaptive) comparison-based $(t, \varepsilon, 2\delta)$-tester for $\mathcal{P}$.*

Our main lower bound theorem is stated next. In the light of the previous discussion, it implies Theorem 1.2.

▶ **Theorem 3.2.** *Any nonadaptive comparison-based tester for unateness of functions* $f : \{0,1\}^d \to \mathbb{R}$ *must make* $\Omega(d \log d)$ *queries.*

By Theorem 3.1 and Yao's minimax principle [34], it suffices to prove the lower bound for deterministic, nonadaptive, comparison-based testers over a known distribution of functions. It may be useful for the reader to recall the sketch of the main ideas given in Section 1.1.1. For convenience, assume d is a power of 2 and let $d' := d + \log_2 d$. We will focus on proving the lower bound for functions $h : \{0,1\}^{d'} \to \mathbb{R}$, as $d \log d = \Theta(d' \log d')$.

3.1 The Hard Distributions

We partition $\{0,1\}^{d'}$ into d subcubes based on the $\log_2 d$ most significant bits. Specifically, for $i \in [d]$, the i^{th} subcube is defined as $C_i := \{x \in \{0,1\}^{d'} : \mathsf{val}(x_{d'} x_{d'-1} \cdots x_{d+1}) = i-1\}$, where $\mathsf{val}(z_p z_{p-1} \dots z_1) := \sum_{i=1}^{p} z_i 2^{i-1}$ is the integer equivalent of the binary string $z_p z_{p-1} \dots z_1$.

Let $m := d$. We denote the set of indices of the subcube by $[m]$ and the set of dimensions by $[d]$. We use $i, j \in [m]$ to index subcubes and $a, b \in [d]$ to index dimensions. We define a series of random variables, where each subsequent variable may depend on the previous ones:

- k: a number picked uniformly at random from $\{1, 2, \dots, \frac{1}{2}\log_2 d\}$.
- R: a uniformly random subset of $[d]$ of size 2^k.
- r_i: for each $i \in [m]$, r_i is picked from R uniformly and independently at random.
- α_b: for each $b \in [d]$, α_b is picked from $\{-1, +1\}$ uniformly and independently at random.
- β_i: for each $i \in [m]$, β_i is picked from $\{-1, +1\}$ uniformly and independently at random.

We denote by $\boldsymbol{S}$ the tuple $(k, R, \{r_i\})$, also referred to as the *shared randomness*. We use $\boldsymbol{T}$ to refer to the entire set of random variables. Given $\boldsymbol{T}$, define the following functions:

$$f_{\boldsymbol{T}}(x) := \sum_{b \in [d'] \setminus R} x_b 3^b + \alpha_{r_i} \cdot x_{r_i} 3^{r_i}, \text{ where } i \text{ is the subcube with } x \in C_i.$$

$$g_{\boldsymbol{T}}(x) := \sum_{b \in [d'] \setminus R} x_b 3^b + \beta_i \cdot x_{r_i} 3^{r_i}, \text{ where } i \text{ is the subcube with } x \in C_i.$$

The distribution **Yes** generates $f_{\boldsymbol{T}}$ and the distribution **No** generates $g_{\boldsymbol{T}}$.

In all cases, the function restricted to any subcube C_i is linear. Consider some dimension $b \in R$. There can be numerous r_i's that are equal to b. For $f_{\boldsymbol{T}}$, in all of these subcubes, the coefficient of x_{r_i} has the same sign, namely α_{r_i}. For $g_{\boldsymbol{T}}$, the coefficient β_i is potentially different, as it depends on the actual subcube. We write $f \sim \mathcal{D}$ to denote that f is sampled from distribution $\mathcal{D}$.

▶ **Claim 3.3.** *Every function* $f \in \mathsf{supp}(\mathbf{Yes})$ *is unate. A function* $g \sim \mathbf{No}$ *is* $\frac{1}{8}$*-far from unate with probability at least* $9/10$.

3.2 From Functions to Signed Graphs that are Hard to Distinguish

For convenience, denote $x \prec y$ if $\mathsf{val}(x) < \mathsf{val}(y)$. Note that $\prec$ forms a total ordering on $\{0,1\}^{d'}$. Given $x \prec y \in \{0,1\}^{d'}$ and a function $h : \{0,1\}^{d'} \to \mathbb{R}$, define $\mathsf{sgn}_h(x, y)$ to be 1 if $h(x) < h(y)$, 0 if $h(x) = h(y)$, and -1 if $h(x) > h(y)$.

Any deterministic, nonadaptive, comparison-based tester is defined as follows: It makes a set of queries Q and decides whether or not the input function h is unate depending on the $\binom{|Q|}{2}$-comparisons in Q. More precisely, for every pair $(x, y) \in Q \times Q$, $x \prec y$, we insert an edge labelled with $\mathsf{sgn}_h(x, y)$. Let this signed graph be called G_h^Q. Any nonadaptive,

comparison-based algorithm can be described as a partition of the universe of all signed graphs over Q into $\mathcal{G}_Y$ and $\mathcal{G}_N$. The algorithm accepts the function h iff $G_h^Q \in \mathcal{G}_Y$.

Let $\boldsymbol{G}_Y^Q$ be the distribution of the signed graphs G_h^Q when $h \sim \mathbf{Yes}$. Similarly, define $\boldsymbol{G}_N^Q$ when $h \sim \mathbf{No}$. Our main technical theorem is Theorem 3.4, which is proved in Section 3.3.

▶ **Theorem 3.4.** *For small enough $\delta > 0$ and large enough d, if $|Q| \leq \delta d \log d$, then $\|\boldsymbol{G}_Y^Q - \boldsymbol{G}_N^Q\|_{\mathrm{TV}} = O(\delta)$.*

The proof of Theorem 3.4 is naturally tied to the behavior of sgn_h. Ideally, we would like to say that $\mathsf{sgn}_h(x, y)$ is almost identical regardless of whether $h \sim \mathbf{Yes}$ or $h \sim \mathbf{No}$. Towards this, we determine exactly the set of pairs (x, y) that potentially differentiate **Yes** and **No**.

▶ **Claim 3.5.** *For all $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$, for all $x \in C_i$ and $y \in C_j$ such that $i < j$, we have $\mathsf{sgn}_h(x, y) = 1$.*

Thus, comparisons between points in different subcubes reveal no information about which distribution h was generated from. Thus the "interesting" pairs that can distinguish whether $h \sim \mathbf{Yes}$ or $h \sim \mathbf{No}$ must lie in the same subcube. The next claim shows a further criterion that is needed for a pair to be interesting. We first define another notation.

▶ **Definition 3.6.** For any setting of the shared randomness $\boldsymbol{S}$, subcube C_i, and points $x, y \in C_i$, we define $t^i_{\boldsymbol{S}}(x, y)$ to be the most significant coordinate of difference (between x, y) in $([d] \setminus R) \cup \{r_i\}$.

Note that $\boldsymbol{S}$ determines R and $\{r_i\}$. For any $\boldsymbol{T}$ that extends $\boldsymbol{S}$ and any function, the restriction to C_i is unaffected by the coordinates in $R \setminus r_i$. Thus, $t^i_{\boldsymbol{S}}(x, y)$ is the first coordinate of difference that is influential in C_i.

▶ **Claim 3.7.** *Fix some $\boldsymbol{S}$, subcube C_i, and points $x, y \in C_i$. Let $c = t^i_{\boldsymbol{S}}(x, y)$, and assume $x \prec y$. For any $\boldsymbol{T}$ that extends $\boldsymbol{S}$:*

- *If $c \neq r_i$, then $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y) = \mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y) = 1$.*
- *If $c = r_i$, $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y) = \alpha_c$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y) = \beta_i$.*

3.3 Proving Theorem 3.4: Good and Bad Events

For a given Q, we first identify certain "bad" values for $\boldsymbol{S}$, on which Q could potentially distinguish between $f_{\boldsymbol{S}}$ and $g_{\boldsymbol{S}}$. We will prove that the probability of a bad $\boldsymbol{S}$ is small for a given Q. Furthermore, we show that Q cannot distinguish between $f_{\boldsymbol{S}}$ and $g_{\boldsymbol{S}}$ for any good $\boldsymbol{S}$. We set up some definitions.

▶ **Definition 3.8.** Given a pair (x, y), define $\mathsf{cap}(x, y)$ to be the 5 most significant coordinates[2] in which they differ. We say (x, y) *captures* these coordinates. For any set $S \subseteq \{0, 1\}^{d'}$, define $\mathsf{cap}(S) := \bigcup_{x,y \in S} \mathsf{cap}(x, y)$ to be the coordinates captured by the set S.

Fix any Q. We set $Q_i := Q \cap C_i$. We define two bad events for $\boldsymbol{S}$.

- Abort Event $\mathcal{A}$: There exist $x, y \in Q$ with $\mathsf{cap}(x, y) \subseteq R$.
- Collision Event $\mathcal{C}$: There exist $i, j \in [d]$ with $r_i = r_j$, $r_i \in \mathsf{cap}(Q_i)$ and $r_j \in \mathsf{cap}(Q_j)$.

If the abort event doesn't occur, then for any pair (x, y), the sign $\mathsf{sgn}_h(x, y)$ is determined by $\mathsf{cap}(x, y)$ for any $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$. The heart of the analysis lies in Theorem 3.9, which states that the bad events happen rarely. Theorem 3.9 is proved in Section 3.4.

[2] There is nothing special about the constant 5. It just needs to be sufficiently large.

▶ **Theorem 3.9.** *If* $|Q| \leq \delta d \log d$*, then* $\Pr[\mathcal{A} \cup \mathcal{C}] = O(\delta)$.

When neither the abort nor the collision events happen, we say $\boldsymbol{S}$ is good for Q. Next, we show that conditioned on a good $\boldsymbol{S}$, the set Q cannot distinguish $f \sim \mathbf{Yes}$ from $g \sim \mathbf{No}$.

▶ **Lemma 3.10.** *For any signed graph* G *over* Q*,*

$$\Pr_{f \sim \mathbf{Yes}}[G_f^Q = G | \boldsymbol{S} \text{ is good}] = \Pr_{g \sim \mathbf{No}}[G_g^Q = G | \boldsymbol{S} \text{ is good}].$$

Proof Sketch. As stated above, when the abort event doesn't happen, the sign $\mathsf{sgn}_h(x, y)$ is determined by $\mathsf{cap}(x, y)$ for any $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$. Furthermore, a pair (x, y) has a possibility of distinguishing (that is, the pair is interesting) only if $x, y \in C_i$ and $r_i \in \mathsf{cap}(x, y)$. Focus on such interesting pairs. For such a pair, both $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y)$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y)$ are equally likely to be $+1$ or -1. Therefore, to distinguish, we would need two interesting pairs, $(x, y) \in C_i$ and $(x', y') \in C_j$ with $i \neq j$. Note that, when $g \sim \mathbf{No}$, the signs $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y)$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x', y')$ are independently set, whereas when $f \sim \mathbf{Yes}$, the signs are either the same when $r_i = r_j$, or independently set. But if the collision event doesn't occur, we have $r_i \neq r_j$ for interesting pairs in different subcubes. Therefore, the probabilities are the same. ◀

Now, we are armed to prove Theorem 3.4.

Proof of Theorem 3.4. Given any subset of signed graphs, $\mathcal{G}$, it suffices to upper bound

$$\left| \Pr_{f \sim \mathbf{Yes}}[G_f^Q \in \mathcal{G}] - \Pr_{f \sim \mathbf{No}}[G_f^Q \in \mathcal{G}] \right| \leq sum_{\text{good } \boldsymbol{S}} \left| \Pr[\boldsymbol{S}] \cdot \left(\Pr_{f \sim \mathbf{Yes}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] - \Pr_{f \sim \mathbf{No}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] \right) \right|$$
$$+ \sum_{\text{bad } \boldsymbol{S}} \left| \Pr[\boldsymbol{S}] \cdot \left(\Pr_{f \sim \mathbf{Yes}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] - \Pr_{f \sim \mathbf{No}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] \right) \right|.$$

The first term of the RHS is 0 by Lemma 3.10. The second term is at most the probability of bad events, which is $O(\delta)$ by Theorem 3.9. ◀

3.4 Bounding the Probability of Bad Events: Proof of Theorem 3.9

We prove Theorem 3.9 by individually bounding $\Pr[\mathcal{A}]$ and $\Pr[\mathcal{C}]$.

▶ **Lemma 3.11.** *If* $|Q| \leq \delta d \log d$*, then* $\Pr[\mathcal{A}] \leq d^{-1/4}$.

Proof. Fix any choice of k (in $\boldsymbol{S}$). For any pair of points $x, y \in Q$, we have $\Pr[\mathsf{cap}(x, y) \subseteq R] \leq (\frac{2^k}{d-5})^5$. Since $d - 5 \geq d/2$ for all $d \geq 10$ and $k \leq (\log_2 d)/2$, the probability is at most $32d^{-5/2}$. For a large enough d, a union bound over all pairs in $Q \times Q$, which are at most $d^2 \log^2 d$ in number, completes the proof. ◀

The collision event is more challenging to bound. Bounding it is the heart of the lower bound. We start by showing that, if each Q_i captures few coordinates, then the collision event has low probability. A critical point is the appearance of $d \log d$ in this bound.

▶ **Lemma 3.12.** *If* $\sum_i |\mathsf{cap}(Q_i)| \leq M$*, then* $\Pr[\mathcal{C}] = O\left(\frac{M}{d \log d}\right)$.

Proof. For any $r \in [d]$, define $A_r := \{j : r \in \mathsf{cap}(Q_j)\}$ to be the set of indices of Q_j's that capture coordinate r. Let $a_r := |A_r|$. Define $n_\ell := |\{r : a_r \in (2^{\ell-1}, 2^\ell]\}|$. Observe that $\sum_{\ell \leq \log_2 d} n_\ell 2^\ell \leq 2 \sum_{r \in [d]} a_r \leq 2M$.

Fix k. For $r \in [d]$, we say the event $\mathcal{C}_r$ occurs if (a) $r \in R$, and (b) there exists $i, j \in [d]$ such that $r_i = r_j = r$, and $r_i \in \mathtt{cap}(Q_i)$ and $r_j \in \mathtt{cap}(Q_j)$. By the union bound, $\Pr[\mathcal{C}|k] \leq \sum_{r=1}^{d} \Pr[\mathcal{C}_r|k]$.

Let us now compute $\Pr[\mathcal{C}_r|k]$. Only sets Q_j's with $j \in A_r$ are of interest, since the others do not capture r. Event $\mathcal{C}_r$ occurs if at least two of these sets have $r_i = r_j = r$. Hence,

$$\begin{aligned}\Pr[\mathcal{C}_r|k] &= \Pr[r \in R] \cdot \Pr[\exists i, j \in A_r : r_i = r_j = r \mid r \in R] \\ &= \frac{2^k}{d} \cdot \sum_{c \geq 2} \binom{a_r}{c} \left(\frac{1}{2^k}\right)^c \left(1 - \frac{1}{2^k}\right)^{a_r - c}. \end{aligned} \tag{1}$$

A fixed r is in R with probability $\binom{d-1}{2^k-1}/\binom{d}{2^k} = \frac{2^k}{d}$. Given that $|R| = 2^k$, the probability that $r_i = r$ is precisely 2^{-k}.

If $a_r \geq \frac{2^k}{4}$, then we simply upper bound (1) by $\frac{2^k}{d}$. For $a_r < \frac{2^k}{4}$, we upper bound (1) by

$$\frac{2^k}{d}\left(1 - \frac{1}{2^k}\right)^{a_r} \sum_{c \geq 2} \left(a_r \cdot \frac{1}{2^k} \cdot \left(1 - \frac{1}{2^k}\right)^{-1}\right)^c \leq \frac{2^k}{d} \sum_{c \geq 2} \left(\frac{a_r}{2^{k-1}}\right)^c \leq \frac{8a_r^2}{2^k d}.$$

Summing over all r and grouping according to n_ℓ, we get

$$\Pr[\mathcal{C}|k] \leq \sum_{r=1}^{d} \Pr[\mathcal{C}_r|k] \leq \sum_{r: a_r \geq 2^{k-2}} \frac{2^k}{d} + \frac{8}{d} \sum_{r: a_r < 2^{k-2}} \frac{a_r^2}{2^k} \leq \frac{2^k}{d} \sum_{\ell > k-2} n_\ell + \frac{8}{d} \sum_{\ell=1}^{k-2} n_\ell 2^{2\ell - k}.$$

Averaging over all k, we get

$$\begin{aligned}\Pr[\mathcal{C}] = \frac{2}{\log_2 d} \sum_{k=1}^{(\log_2 d)/2} \Pr[\mathcal{C}|k] \quad &\leq \quad \frac{16}{d \log_2 d} \sum_{k=1}^{(\log_2 d)/2} \left(\sum_{\ell=1}^{k-2} n_\ell 2^{2\ell - k} + \sum_{\ell > k-2} n_\ell 2^k\right) \\ &= \frac{16}{d \log_2 d} \left(\sum_{\ell=1}^{(\log_2 d)/2} n_\ell \sum_{k \geq \ell+2} 2^{2\ell - k} + \sum_{\ell=1}^{\log_2 d} n_\ell \sum_{k < \ell+2} 2^k\right). \end{aligned} \tag{2}$$

Now, $\sum_{k \geq \ell+2} 2^{2\ell-k} \leq 2^\ell$ and $\sum_{k < \ell+2} 2^k \leq 4 \cdot 2^\ell$. Substituting, $\Pr[\mathcal{C}] \leq \frac{80}{d \log_2 d} \sum_{\ell=1}^{\log_2 d} n_\ell 2^\ell \leq \frac{160M}{d \log_2 d}$, proving the lemma. ◀

We are now left to bound $\sum_i |\mathtt{cap}(Q_i)|$. This is done by the following combinatorial lemma.

▶ **Lemma 3.13.** *Let V be a set of vectors over an arbitrary alphabet and any number of dimensions. For any natural number c and $x, y \in V$, let $\mathtt{cap}_c(x, y)$ denote the (set of) first c coordinates at which x and y differ. Then $|\mathtt{cap}_c(V)| \leq c(|V| - 1)$.*

Proof. We construct c different edge-coloured graphs $G_1, \ldots, G_c$ over the vertex set V. For every coordinate $i \in \mathtt{cap}_c(V)$, there must exist at least one pair of vectors x, y such that $i \in \mathtt{cap}_c(x, y)$. Thinking of each $\mathtt{cap}_c(x, y)$ as an ordered set, find a pair (x, y) where i appears "earliest" in $\mathtt{cap}_c(x, y)$. Let the position of i in this $\mathtt{cap}_c(x, y)$ be denoted t. We add edge (x, y) to G_t, and colour it i. Note that the same edge (x, y) cannot be added to G_t with multiple colours, and hence all G_t's are simple graphs. Furthermore, observe that each colour is present only once over all G_t's.

We claim that each G_t is acyclic. Suppose not. Let there be a cycle C and let (x, y) be the edge in C with the smallest colour i. Clearly, $x_i \neq y_i$ since $i \in \mathtt{cap}_c(x, y)$. There must exist another edge (u, v) in C such that $u_i \neq v_i$. Furthermore, the colour of (u, v) is $j > i$. Thus, j is the t^{th} entry in $\mathtt{cap}_c(u, v)$. Note that $i \in \mathtt{cap}_c(u, v)$ and must be the s^{th} entry for some $s < t$. But this means that the edge (u, v) coloured i should be in G_s, contradicting the presence of $(x, y) \in G_t$. ◀

We wrap up the bound now.

▶ **Lemma 3.14.** *If* $|Q| \leq \delta d \log d$, *then* $\Pr[\mathcal{C}] = O(\delta)$.

Proof. Lemma 3.13 applied to each Q_i, yields $\sum_i |\mathtt{cap}(Q_i)| \leq 5|Q_i| = 5|Q|$. An application of Lemma 3.12 completes the proof. ◀

Acknowledgments. We thank Oded Goldreich for useful discussions and Meiram Murzabulatov for participation in initial discussions on this work.

References

1 Nir Ailon and Bernard Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Inf. Comput.*, 204(11):1704–1717, 2006.

2 Roksana Baleshzar, Deeparnab Chakrabarty, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and C. Seshadhri. Optimal unateness testers for real-valued functions: Adaptivity helps. *Electronic Colloquium on Computational Complexity (ECCC)*, 2017, 2017. Also appeared as arXiv report 1703.05199 (`https://arxiv.org/abs/1703.05199`). URL: `https://eccc.weizmann.ac.il/report/2017/049/`.

3 Tugkan Batu, Ronitt Rubinfeld, and Patrick White. Fast approximate PCPs for multidimensional bin-packing problems. *Inf. Comput.*, 196(1):42–56, 2005.

4 Aleksandrs Belovs and Eric Blais. Quantum algorithm for monotonicity testing on the hypercube. *Theory of Computing*, 11:403–412, 2015.

5 Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 1021–1032, 2016.

6 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 164–173, 2014.

7 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012.

8 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

9 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings, IEEE Conference on Computational Complexity (CCC)*, pages 309–320, 2014.

10 Jop Briët, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

11 Deeparnab Chakrabarty. Monotonicity testing. In *Encyclopedia of Algorithms*, pages 1352–1356. Springer, 2016.

12 Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1809–1828, 2015.

13 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 2013.

14 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.

15 Deeparnab Chakrabarty and C. Seshadhri. An $o(n)$ monotonicity tester for boolean functions over the hypercube. *SIAM J. Comput.*, 45(2):461–472, 2016.

16 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $O(n^{1/2})$ non-adaptive queries. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 519–528, 2015.

17 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 286–295, 2014.

18 Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin M. Varma. Erasure-resilient property testing. In *Proceedings, International Colloquium on Automata, Languages and Processing (ICALP)*, pages 91:1–91:15, 2016.

19 Yevgeny Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. *Proceedings, International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 97–108, 1999.

20 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. System Sci.*, 60(3):717–751, 2000.

21 Eldar Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107–116, 2004.

22 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 474–483, 2002.

23 Oded Goldreich. Introduction to property testing (working draft), 2015. URL: `http://www.wisdom.weizmann.ac.il/~oded/PDF/pt-v1.pdf`.

24 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20:301–337, 2000.

25 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

26 Shirley Halevy and Eyal Kushilevitz. Distribution-free property-testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007.

27 Shirley Halevy and Eyal Kushilevitz. Testing monotonicity over graph products. *Random Struct. Algorithms*, 33(1):44–67, 2008.

28 Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM J. Comput.*, 42(2):700–731, 2013.

29 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric type theorems. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 52–58, 2015.

30 Subhash Khot and Igor Shinkar. An $\widetilde{O}(n)$ queries adaptive tester for unateness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016*, pages 37:1–37:7, 2016.

31 Eric Lehman and Dana Ron. On disjoint chains of subsets. *J. Combin. Theory Ser. A*, 94(2):399–404, 2001.

32 Sofya Raskhodnikova. Testing if an array is sorted. In *Encyclopedia of Algorithms*, pages 2219–2222. Springer, 2016.

33 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

34 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

Sublinear Random Access Generators for Preferential Attachment Graphs

Guy Even[1], Reut Levi[2], Moti Medina[3], and Adi Rosén[*4]

1 **Tel Aviv University, Tel Aviv, Israel**
guy@eng.tau.ac.il
2 **MPI for Informatics, Saarland Informatics Campus, Saarbrücken, Germany**
reuti.levi@gmail.com
3 **MPI for Informatics, Saarland Informatics Campus, Saarbrücken, Germany**
moti.medina@gmail.com
4 **CNRS and Université Paris Diderot, Paris, France**
adiro@liafa.univ-paris-diderot.fr

Abstract

We consider the problem of sampling from a distribution on graphs, specifically when the distribution is defined by an evolving graph model, and consider the time, space and randomness complexities of such samplers.

In the standard approach, the whole graph is chosen randomly according to the randomized evolving process, stored in full, and then queries on the sampled graph are answered by simply accessing the stored graph. This may require prohibitive amounts of time, space and random bits, especially when only a small number of queries are actually issued. Instead, we propose to generate the graph on-the-fly, in response to queries, and therefore to require amounts of time, space, and random bits which are a function of the actual number of queries.

We focus on two random graph models: the Barabási-Albert Preferential Attachment model (BA-graphs) [3] and the random recursive tree model [24]. We give on-the-fly generation algorithms for both models. With probability $1 - 1/\text{poly}(n)$, each and every query is answered in $\text{polylog}(n)$ time, and the increase in space and the number of random bits consumed by any single query are both $\text{polylog}(n)$, where n denotes the number of vertices in the graph.

Our results show that, although the BA random graph model is defined by a sequential process, efficient random access to the graph's nodes is possible. In addition to the conceptual contribution, efficient on-the-fly generation of random graphs can serve as a tool for the efficient simulation of sublinear algorithms over large BA-graphs, and the efficient estimation of their performance on such graphs.

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

Keywords and phrases local computation algorithms, preferential attachment graphs, random recursive trees, sublinear algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.6

1 Introduction

Consider a Markov process in which a sequence $\{S_t\}_t$ of states, $S_t \in \mathcal{S}$, evolves over time $t \geq 1$. Suppose there is a set $\mathcal{P}$ of predicates defined over the state space $\mathcal{S}$. Namely, for every predicate $P \in \mathcal{P}$ and state $S \in \mathcal{S}$, the value of $P(S)$ is well defined. A query is a pair

* Research supported in part by ANR projet RDAM.

(P, t) and the answer to the query is $P(S_t)$. In the general case, answering a query (P, t) requires letting the Markov process run for t steps until S_t is generated. In this paper we are interested in ways to reduce the dependency, on t, of the computation time time, the memory space, and the number of used random bits, required to answer a query (P, t).

We focus on the case of generative models for random graphs, and in particular, on the Barabási-Albert Preferential Attachment model [3] (which we call BA-graphs), on the equivalent linear evolving copying model of Kumar et al. [10], and on the random recursive tree model [24]. The question we address is whether one can design a randomized *on-the-fly* graph generator that answers adjacency list queries of BA-graphs (or random recursive trees), without having to generate the complete graph. Such a generator outputs answers to adjacency list queries as if it first selected the whole graph at random (according the appropriate distribution) and then answered the queries based on the samples graph.

We are interested in the following resources of a graph generator: (1) the number of random bits consumed per query, (2) the running time per query, and (3) the increase in memory space per query.

Our main result is a randomized on-the-fly graph generator for BA-graphs over n vertices that answers adjacency list queries. The generated graph is sampled according to the distribution defined for BA-graphs over n vertices, and the complexity upper bounds that we prove hold with probability $1 - 1/\text{poly}(n)$. That is, with probability $1 - 1/\text{poly}(n)$ each and every query is answered in $\text{polylog}(n)$ time, and the increase in space, and the number of random bits consumed during that query are $\text{polylog}(n)$. Our result refutes (definitely for $\text{polylog}(n)$ queries) the recent statement of Kolda et al. [9] that: "The majority of graph models add edges one at a time in a way that each random edge influences the formation of future edges, making them inherently serial and therefore unscalable. The classic example is Preferential Attachment, but there are a variety of related models..."

We remark that the entropy of the edges in BA-graphs is $\Theta(\log n)$ per edge in the second half of the graph [23]. Hence it is not possible to consume a sublogarithmic number of random bits per query in the worst case if one wants to sample according to the BA-graph distribution. Similarly, to insure consistency (i.e., answer the same query twice in the same way) one must use $\Omega(\log n)$ space per query.

From a conceptual point of view, the main ingredient of our result are techniques to "invert" the sequential process where each new vertex randomly selects its "parent" in the graph among the previous vertices. Instead, vertices randomly select their "children" among the "future" vertices, while maintaining the same probability distribution as if each child picked "in the future" its parent. We apply these techniques in the related model of random recursive trees [24] (also used within the evolving copying model [10]), and use them as a building block for our main result for BA-graphs.

Due to space limitations, some of the proofs are omitted from this extended abstract.

Related work. A linear time randomized algorithm for efficiently generating BA-graphs is given in Betagelj and Brandes [4]. See also Kumar et al. [10] and Nobari et al. [18]. A parallel algorithm is given in Alam et al. [1]. See also Yoo and Henderson [25]. An external memory algorithm was presented by Meyer and Peneschuck [16]. Generating huge random objects while using "small" amounts of randomness was studied by Goldreich, Goldwasser and Nussboim [8]. Mansour et al. [14] consider local generation of bipartite graphs in the context of local simulation of Balls into Bins online algorithms.

Applications. One reason for generating large BA-graphs is to simulate algorithms over them. Such algorithms often access only small portions of the graphs. In such instances, it is

wasteful to generate the whole graph. An interesting example is sublinear approximation algorithms [20, 26, 17, 19] which probe a constant number of neighbors. In addition, local computation algorithms probe a small number of neighbors to provide answers to optimization problems such as maximal independent sets and approximate maximum matchings [6, 7, 21, 22, 2, 14, 15, 11, 12, 13]. Support of adjacency list queries is especially useful for simulating (partial) DFS and BFS over graphs.

2 Preliminaries

Let $V_n \triangleq \{v_1, \ldots, v_n\}$. Let $G = (V_n, E)$ denote a directed graph on n nodes.[1] We refer to the endpoints of a directed edge (u, v) as the *head* v and the *tail* u. Let $\deg(v_i, G)$ denote the *degree* of the vertex v_i in G (both incoming and outgoing edges). Similarly, let $deg_{in}(v_i, G)$ and $deg_{out}(v_i, G)$ denote the in-degree and out-degree, respectively, of the vertex v_i in G.

In the sequel, when we say that an event occurs *with high probability* (or *w.h.p*) we mean that it occurs with probability at least $1 - \frac{1}{n^c}$, for some constant c.

For ease of presentation, we extensively use in the algorithms arrays of size n. However, in order to keep the space complexity low, we implement these arrays by means of balanced search trees, with keys in $[1, n]$. Thus, the space used by the "arrays" is the number of keys stored. The time complexities that we give are therefore to be multiplied by a factor of $O(\log n)$.

3 Queries and On-the-Fly Generators

Consider an undirected graph $G = (V_n, E)$, where $V_n = \{v_1, \ldots, v_n\}$. Slightly abusing notation, we sometimes consider and denote node v_i as the integer number i and so we have a natural order on the nodes. The access to the graph is done by means of a user-query `BA-next-neighbor` : $[1, n] \to [1, n + 1]$, where $n + 1$ denotes "no additional neighbor". We number the queries according to the order they are issued, and call this number the *time* of the query. Let q_t be the node on which the query at time t was issued, i.e, at time t the query `BA-next-neighbor`(q_t) is issued by the user. For each node $v \in V$ and any time t, let $last_t(j)$ be the largest numbered node which was previously returned as the value of `BA-next-neighbor`(j), or 0 if no such query was issued before time t. That is, $last_t(v) = \min\{0, \min_{t' < t}\{\texttt{BA-next-neighbor}(q_{t'}) | q_{t'} = v\}$. At time t the query `BA-next-neighbor`(v) returns $\arg\min_{i > last_t(j)}\{(i, j) \in E\}$, or $n+1$ if no such i exists. When the implementation of the query has access to a data structure holding the whole of E, then the implementation of `BA-next-neighbor` is straightforward just by accessing this data structure.

An on-the-fly graph generator is an algorithm that gives access to a graph by means of the `BA-next-neighbor` query defined above, but itself does not have access to a data structure that encodes the whole graph. Instead, in response to the queries issued by the user, the generator modifies its internal data structure (a.k.a state), which is initially some empty (constant) state. The generator must ensure however that its answers are consistent with some graph G. An on-the-fly graph generator for a given distribution on a family of graphs (such as the family of Preferential Attachment graphs on n nodes) must in addition

[1] Preferential attachment graphs are usually presented as undirected graphs. For convenience of discussion we orient each edge from the high index vertex to the low index vertex, but the graphs we consider remain undirected graphs.

ensure that it samples the graphs according to the required distribution. That is, its answers to a sequence of queries must be *distributed identically* to those returned when a graph was first sampled (according to the desired distribution), stored, and then accessed (See Definition 16 and Theorem 17).

4 Random Graph Models

Preferential attachment [3]. We restrict our attention to the case in which each vertex is connected to the previous vertices by a single edge (i.e., $m = 1$ in the terminology of [3]). We thus denote the random process that generates a graph over V_n according to the preferential attachment model by BA_n. The random process BA_n generates a sequence of n directed edges $E_n \triangleq \{e_1, \ldots, e_n\}$, where the tail of e_i is v_i, for every $i \in [1, n]$. (We abuse notation and let $BA_n = (V_n, E_n)$ also denote the graph generated by the random process.) We refer to the head of e_i as the *parent* of v_i.

The process BA_n draws the edges sequentially starting with the self-loop $e_1 = (v_1, v_1)$. Suppose we have selected BA_{j-1}, namely, we have drawn the edges $e_1, \ldots, e_{j-1}$, for $j > 1$. The edge e_j is drawn such its head is node v_i with probability $\frac{\deg(v_i, G)}{2(j-1)}$.

Note that the out-degree of every vertex in (the directed graph representation of) BA_n is exactly one, with only one self-loop in v_1. Hence BA_n (without the self-loop) is an in-tree rooted at v_1.

Evolving copying model [10]. Let Z_n denote the evolving copying model with out-degree $d = 1$ and copy factor $\alpha = 1/2$. As in the case of BA_n, the process Z_n selects the edges $E'_n = \{e'_1, \ldots, e'_n\}$ one-by-one starting with a self-loop $e'_1 = (v_1, v_1)$. Given the graph $Z_{n-1} = (V_n, E'_n)$, the next edge e'_n emanates from v_n. The head of edge e'_n is chosen as follows. Let $b_n \in \{0, 1\}$ be an unbiased random bit. Let $u(n) \in [1, n-1]$ be a uniformly distributed random variable (the random variables $b_1, \ldots, b_n$ and $u(1), \ldots, u(n)$ are all pairwise independent.) The head v_i of e'_n is determined as follows: $head(e'_n) \triangleq u(n)$, if $b_n = 1$; and $head(e'_n) \triangleq head(e_{u(n)})$, if $b_n = 0$.

Random recursive tree model [24]. If we eliminate from the evolving copying model the bits b_i and the "copying effect" we get a model where each new node n is connected to one of the previous nodes, chosen uniformly at random. This is the extensively studied (random) recursive tree model [24].

We now relate the various models. Proof omitted from this extended abstract.

▶ **Claim 1** ([1]). *The random graphs BA_n and Z_n are identically distributed.*

We use the following claim in the sequel.

▶ **Claim 2** (cf. [5], Thm. 6.12 and Thm. 6.32). *Let T be a rooted directed tree on n nodes denoted $1, \ldots, n$, and where node 1 is the root of the tree. If the head of the edge emanating from node $j > 1$ is uniformly distributed among the nodes in $[1, j-1]$, then, with high probability, the following two properties hold:*

1. *The maximum in-degree of a node in the tree is $O(\log n)$.*
2. *The height of the tree is $O(\log n)$.*

5 The Pointers Tree

We now consider a graph inspired by the the random recursive tree model [24] and the evolving copying model [10]. Each vertex i has a variable $u(i)$ that is uniformly distributed

over $[1, i-1]$, and can be viewed as a directed edge (or pointer) from i to $u(i)$. We denote this random rooted directed in-tree by UT. Let $u^{-1}(j)$ denote the set $\{i : u(i) = j\}$. We refer to the set $u^{-1}(i)$ as the *u-children* of i and to $u(i)$ as the *u-parent* of i. In conjunction with each pointer, we keep a flag indicating whether this pointer is to be used as a `dir` (direct) pointer or as a `rec` (recursive) pointer. We thus use the directed pointer tree to represent a graph in the evolving copying model (which is equivalent, when the flag of each pointer is equality distributed between `rec` and `dir`, to the BA model).

In this section we consider the subtask of giving access to a random UT, together with the flags of each pointer. Ignoring the flags, this section thus gives an on-the-fly random access generator for the extensively studied model of random recursive trees (cf. [24]). We define the following queries.

- $(i, flag) \leftarrow$ `parent`(j): i is the parent of j in the tree, and $flag$ is the associated flag.
- $i \leftarrow$ `next-child-tp`$(j, k, type)$, where $k \geq j$: i is the least numbered node $i > k$ such that the parent of i is j and the flag of that pointer is of type "type". If no such node exists then i is $n + 1$.

The "ideal" way to implement this task is to go over all n nodes, and for each node j (1) uniformly at random choose its parent in $[1, j-1]$, (2) uniformly at random chose the associated flag in $\{$`dir`, `rec`$\}$. Then store the pointers and flags, and answer the queries by accessing this data structure.

In this section we give an *on-the-fly* generator that answers the above queries. We start with some notations. We say that j is *exposed* if $u(j) \neq$ nil (initially all pointers $u(j)$ are set to nil). We denote the set of all exposed vertices by F. We say that j is *directly* exposed if $u(j)$ was set during an invocation of `next-child-tp`$(i, \cdot, \cdot)$. We say that j is *indirectly* exposed if $u(j)$ was determined during an invocation of `parent`(j). As a result of answering and processing `next-child-tp` and `parent` queries, the on-the-fly generator commits to various decisions (e.g., prefixes of adjacency lists). These commitments include edges but also non-edges (i.e., vertices that can no longer serve as $u(j)$ for a certain j). For a node i, $front(i)$ denotes the largest value (node) $k \in [1, n+1]$ such that k was returned by a `next-child-tp`$(i, \cdot, \cdot)$ query, and nil if no such returned value exists. Observe that $front(i) = k$ implies that (1) $u(k) = i$; and (2) we know already for each node $j \in [j+1, k-1]$ if $u(j) = i$ or not. We denote - roughly speaking - the set of vertices that cannot serve as u-parents of j by $not\text{-}u\text{-}parent\text{-}candidate(j)$, the nodes that can still be u-parents of j by $\Phi(j)$, and their number by $\varphi(j) = |\Phi(j)|$. The formal definitions are:

$$
\begin{aligned}
not\text{-}u\text{-}parent\text{-}candidate(j) &\triangleq \{i \in [1, j-1] : front(i) \geq j\} \ , \\
\Phi(j) &\triangleq [1, j-1] \setminus not\text{-}u\text{-}parent\text{-}candidate(j) \ , \\
\varphi(j) &\triangleq |\Phi(j)| \ .
\end{aligned}
$$

5.1 An efficient implementation of `next-child`

We first shortly discuss the challenges on the way to an efficient implementation of `next-child`. Observe that before the first `next-child`(j) query, for a given j, is issued, the probability for any $x > j$ to be a u-child of j is $1/\varphi(x)$, because all nodes $x' < x$ can still be the u-parent of x. But once `next-child`$(\cdot)$ queries are issued, this may no longer be the case. For example, if $x > front(j')$, then, even if the u-parent of x is not yet determined, j' is no longer an option to be the u-parent of x. This renders the calculation of $\mathbf{Pr}[u(x) = j]$ more complicated and more computation-time consuming, which renders the process of selecting the next child of a node j non-efficient. In the rest of this section we show how to overcome these

difficulties and give a procedure that selects the next child, with the appropriate probability distribution, using polylog(n) random bits and in polylog(n) time, and while increasing the space by polylog(n). This procedure will be at the heart of our efficient implementation of `next-child`.

The efficient implementation of `next-child` (and of `parent`) makes use of the following data structures.

- An array of length n, $u(j)$
- An array of length n, *type*(j)
- An array of length n, *front*(j) (We also maintain an array $front^{-1}(i)$ with the natural definition.)
- An array of n balanced search trees, called `child`(j), each holding a set of nodes $i > j$ such that $u(i) = j$ (not necessarily all such nodes). For technical reasons we initiate all trees `child`(j) with $n + 1 \in$ `child`(j).
- A number of additional data structures that are implicit in the listing, described and analyzed in the sequel.

In the implementation we maintain the following two invariant.

▶ **Invariant 3.** *For every node j, the first* ***next-child-tp***$(j, \cdot, \cdot)$ *query is always preceded by a* ***parent***(j) *query.*

We will use this invariant to infer that *front*$(j) \neq$ nil implies that $u(j) \neq$ nil. One can easily maintain this invariant by introducing a `parent`(j) query as the first step of the implementation of the `next-child-tp`$(j, \cdot, \cdot)$ query (for technical reasons we do that in a lower-level procedure `next-child`.)

▶ **Invariant 4.** *For every vertex j, front*$(j) \neq$ nil *implies that front*$(front(j)) \neq$ nil*.*

The second invariant is maintained by issuing an "internal" `next-child`$(front(j), front(j))$ query whenever *front*(j) is updated. This is done recursively, the base of the recursion being node $n+1$. Let $front^{-1}(j)$ denote the vertex i such that *front*$(i) = j$, if such a vertex i exists; otherwise $front^{-1}(j) =$ nil. We get that if $front^{-1}(j) \neq$ nil, then $u(j) \neq$ nil.

▶ **Definition 5.** At a given time t, and for any node j, let $\Phi(j)$ and $\phi(j)$ be defined as follows: $\Phi(j) \triangleq \{i \mid i < j$ `and` (*front*$(i) < j$ `or` *front*$(i) =$ nil$)\}$, and $\phi(j) = |\Phi(j)|$.

The following lemma gives properties of the series $\{\Phi(x)\}_x$. Proof omitted.

▶ **Lemma 6.** *For every $x \in [1, n-1]$:*

1. $\Phi(x) \subseteq \Phi(x+1) \subseteq \Phi(x) \cup \{x, front^{-1}(x)\}$.
2. $\Phi(x+1) = \Phi(x)$ *iff* $x \in K$.
3. $\varphi(x+1) - \varphi(x) \leq 1$.

We now describe the implementation of `next-child-tp`$(j, k, type)$ and `next-child`(j). `next-child-tp`$(j, k, type)$ is a loop of `next-child-from`(j, k) until the right type is found, and `next-child-from`(j, k) is essentially a call to `next-child`(j) (see Figure 1). Note that if j does not have children larger than k, then `next-child-from`(j, k) returns $n + 1$.

If *front*$(j) > k$ when `next-child-from`(j, k) is called, then the next child is already fixed and it is just extracted from the data structures. Otherwise, an interval $I = [a, b]$ is defined, and it will contain the answer of `next-child`(j). Let $a =$ *front*$(j) + 1$ if *front*$(j) \neq$ nil; otherwise $a = j + 1$. Let b denote the smallest, larger than *front*(j), indirectly exposed child of j if one exists (i.e., if *front*$(j) \neq$ nil then $b = \min\{\ell > front(j) \mid u(\ell) = j\}$); if no such b

exists then $b = n+1$. By the definition of K, $K \subseteq F$, and no vertex $x \in F \cap [a,b)$ can satisfy $u(x) = j$. Hence, the answer is in $I \setminus (F \setminus \{b\})$.

The next child can be sampled according to the desired distribution in a straightforward way by going sequentially over the vertices in $I \setminus F \setminus \{b\}$, and tossing for each vertex x a coin that has probability $1/\varphi(x)$ to be 1, until indeed one of those coins comes out 1, or all vertices are exhausted (in which case node b is taken as the next child). However, this procedure takes linear time. We denote by $D(x)$, $x \in I \setminus F$, the probability that x is chosen according to the above procedure. In order to start building our efficient implementation for `next-child` we consider the same process, with the same probabilities $1/\varphi(x)$, but this time for $[a,b) \setminus K$, rather than $[a,b) \setminus F$. The vertex on which we stop, denote is x, is a *candidate next u-child*. If $x \in F \setminus K$, then x cannot be a child of j so we proceed in the same way, but with the interval $[x+1, b]$.

We now build our efficient procedure that selects the candidate, without sequentially going over the nodes. To this end, observe that the sequence of probabilities of the coins tossed in the last-described process behaves "nicely". Namely, the probabilities $1/\varphi(x)$, for $x \in [a,b)\setminus K$, form the harmonic sequence starting from $1/\varphi(a)$ and ending in $1/(\varphi(a) + |[a,b) \setminus K| - 1)$. Indeed, Lemma 6 implies that if vertex i is the smallest vertex in $I \setminus K$, then $\varphi(i) = \varphi(a)$ and an increment between $\varphi(x)$ and $\varphi(x+1)$ occurs if and only if $x \notin K$. Let $s = |I \setminus K|$ and let P_q, $0 \leq q \leq s-1$ be the probability that the node of rank q in $I \setminus K$ is chosen as candidate in the sequential procedure defined above. Since $\varphi(x)$ form the harmonic sequence for $x \in [a,b) \setminus K$, we can calculate in $O(1)$ time, for any $0 \leq i \leq s-1$, the probability $P'_i = \sum_{q<i} P_q$ (i.e., a node of some rank q, $q < i$, is chosen). Indeed, for $i = 0$, $P_i = \frac{1}{\varphi(a)}$; for $0 < i < s-1$, $P_i = \frac{1}{\varphi(a)+i} \cdot \prod_{\ell=0}^{i-1} \left(1 - \frac{1}{\varphi(a)+\ell}\right) = \frac{\varphi(a)-1}{(\varphi(a)+i-1)(\varphi(a)+i)}$; and for $i = s-1$, $P_{s-1} = \prod_{\ell=0}^{s-2} \left(1 - \frac{1}{\varphi(a)+\ell}\right) = \frac{\varphi(a)-1}{\varphi(a)+s-2}$. Hence, for $0 \leq i \leq s-1$, $P'_i = 1 - \frac{\varphi(a)-1}{\varphi(a)+(i-1)}$, and for $i = s$, $P'_s = 1$. This allows us to simulate one iteration (i.e., choosing the next *candidate next u-child*) by choosing uniformly at random a single number in $[0,1]$, and then performing a binary search over 0 to $s-1$ to decide what rank h this number "represents". After we randomly chose a rank $h \in [0, s-1]$, h is then mapped to the vertex of rank h in $I \setminus K$, denote it x, and this is the *candidate next u-child*. As before, if $x \in (F \setminus K)$, then x cannot be a child of j so we ignore it and proceed in the same way, this time with the interval $[x+1, b]$. See the pseudo code of of `next-child` and `toss` (Figure 1). We denote by $\hat{D}(x)$, $x \in I \setminus F$ the probability that x is chosen according to this third procedure. See Figure 1 for a formal definition of this procedure.

Observe that this procedure takes $O(\log s)$ time (see Section 5.2 for a formal statement of the time and randomness complexities). We note that we cannot perform this selection procedure in the same time complexity for the set $[a,b) \setminus F$, because we do not have a way to calculate each and every probability P'_i, $i \in [a,b) \setminus F$, in $O(1)$ time.

To conclude the description of the implementation of `next-child`, we give the following lemma which states that the probability distribution on the next child is the same for all three processes described above. The (technical) proof is omitted.

▶ **Lemma 7.** *For all $x \in I \setminus F$, $\hat{D}(x) = D(x)$.*

The implementation of `parent` is straightforward (see Figure 1). However, note that updating the various data structures, while implicit in the listing, is accounted for in the time analysis.

```
1: procedure NEXT-CHILD-TP(j, k, type)
2:     x ← k
3:     repeat
4:         x ← next-child-from(j, x)
5:     until flag(x) = type or x = n + 1
6:     return x
7: end procedure
```

```
1: procedure NEXT-CHILD-FROM(j, k)
2:     If (k ≥ n) return (n + 1)
3:     q ← succ(child(j), k)
4:     if q ≤ front(j) then
5:         return q
6:     else
7:         return next-child(j)
8:     end if
9: end procedure
```

```
1: procedure PARENT(j)
2:     if u(j) = nil then
3:         u(j) ←_R [1, j − 1]
4:         type(j) ←_R {dir, rec}
5:     end if
6:     return (u(j), type(j))
7: end procedure
```

```
1: procedure NEXT-CHILD(j)
2:     (p, t) ← parent(j)
3:     If (front(j) ≥ n) return (n + 1)
4:     a ← { front(j) + 1   if front(j) ≠ nil
           { j + 1          if front(j) = nil
5:     b ← { succ(child(j), front(j))   if front(j) ≠ nil
           { n + 1                      if front(j) = nil
6:     repeat
7:         s ← |[a, b] \ K|
8:         h ← toss(φ(a), s)
9:         x ← the vertex of rank h in [a, b] \ K
10:        if x = b then
11:            return b
12:        else
13:            if u(x) = nil then
14:                u(x) = j
15:                type(x) ←_R {dir, rec}
16:                front(j) ← x
17:                front^{-1}(x) ← j
18:                if (front(x) = nil) next-child(x)
19:                return (x)
20:            else
21:                if u(x) = j then
22:                    front(j) ← x
23:                    front^{-1}(x) ← j
24:                    if (front(x) = nil) next-child(x)
25:                    return(x)
26:                else
27:                    a ← x + 1
28:                end if
29:            end if
30:        end if
31:    until forever
32: end procedure
```

```
1: procedure TOSS(φ, s)
2:     α ← n^c (for some constant c > 1).
3:     Choose uniformly at random an integer H ∈ [0, α]
4:     M ← H · 1/α
5:     Using binary search on [0, s − 1] find y such that P'_y ≤ M < P'_{y+1}
6:                 (where, for 0 ≤ y ≤ s − 1, P'_y = 1 − (φ−1)/(φ+(y−1)), and P'_s = 1)
7:     if (H + 1) 1/α ≤ Pr_{y+1} then
8:         return y
9:     else
10:        α ← α · Π_{y=0}^{s−1}(P'_{y+1} − P'_y)
11:        Choose uniformly at random an integer H ∈ [0, α]
12:        M ← H · 1/α
13:        Using binary search on [0, s − 1] find y such that P'_y ≤ H < P'_{y+1}
14:                (where, for 0 ≤ y ≤ s − 1, P'_y = 1 − (φ−1)/(φ+(y−1)), and P'_s = 1)
15:        return y
16:    end if
17: end procedure
```

Figure 1 Pseudo code of the pointers tree generator.

5.2 Analysis of the pointer tree generator

We first give the following claim that we later use a number of times.

▶ **Lemma 8.** *With high probability, for each and every invocation of* `next-child`*, the size of the recursion tree of that invocation for calls to* `next-child` *is* $O(\log n)$.

Proof. Consider the recursive invocation tree that results from a call to `next-child`. Observe that (1) by the code of `next-child` this tree is in fact a path; and (2) this path corresponds to a path in the pointers tree, where each edge of this tree-path is "discovered" by the corresponding call to `next-child`. That is, the maximum size of an invocation tree of a call of `next-child` is bounded from above by the height of the pointers tree. By Claim 2, with high probability, this is $O(\log n)$. ◀

5.2.1 Data structures and space complexity

The efficient implementation of `next-child` makes use of the following data structures.

- A number of arrays of length n, $u(j)$ and $type(j)$, $front(j)$ and $front^{-1}(j)$, used to store various values for nodes j. Since we implement arrays by means of search trees, the space complexity of each array is $O(m)$, where m is the maximum number of distinct keys stored with a non-null value in that array, at any given time. The time complexity for each operation is $O(\log m) = O(\log n)$.
- For each node j, a balanced binary search tree called $\texttt{child}(j)$, where $\texttt{child}(j)$ stores (some of the known) children of node j. (for technical reasons we define $\texttt{child}(j)$ to always include node $n+1$.) Observe that for each child i stored in one of these trees, $u(i)$ is already determined. Thus, the increase, during a given period, in the space used by the `child` trees is bounded from above by the the number of nodes i for which $u(i)$ got determined during that period. For the time complexity of the operations on these trees we use a coarse standard upper bound of $O(\log n)$.

The listings of the implementations of the various procedures leave *implicit* the maintenance of two data structures, related to the set K and to the computation of $\varphi(\cdot)$:

- A data structure that allows one to retrieve the value of $\varphi(a)$ for a given vertex a. This data structure is implemented by retrieving the cardinality of *not-u-parent-candidate*(a) for a given node a. The latter is equivalent to counting how many nodes $i < a$ have $front(i) \neq \mathsf{nil}$ and $front(i) \geq a$. We use two balanced binary search trees (or order statistics trees) in a specific way and have that by standard implementations of balanced search trees the space complexity is $O(k)$ (and all operations are done in time $O(\log k) = O(\log n)$). Here k denotes the number of nodes i such that $front(i) \neq \mathsf{nil}$. The details of the implementation are omitted from this extended abstract.
- A data structure that allows one to find the vertex of rank h in the ordered set $[a, n+1] \setminus K$. This data structure is implemented by a balanced binary search tree storing the nodes in K, augmented with the queries $rank_K(i)$ (as in an order-statistics tree) as well as $rank_{\bar{K}}(i)$ and $select_{\bar{K}}(s)$, i.e., finding the element of rank s in the complement of K. To find the vertex of rank h in $[a, n+1] \setminus K$ we use the query $select_{\bar{K}}(rank_{\bar{K}}(a) + h)$. The space complexity of this data structure is $O(k)$, and all operations are done in time $O(\log k) = O(\log n)$ or $O(\log^2 k) = O(\log^2 n)$ (for the $select_{\bar{K}}(i)$ query). Here k denotes the number of nodes in K, which is upper bounded by the number of nodes i such that $front(i) \neq \mathsf{nil}$. The details of the implementation are omitted from this extended abstract.

5.2.2 Time complexity

Time complexity of `toss`(φ, s). The time complexity of this procedure is $O(1)$ regardless of whether or not the `if` condition holds or not.

Time complexity of "$x \leftarrow$ the vertex of rank h in $[a, n+1] \setminus K$". This operation is implemented using the data structure defined above, and takes $O(\log^2 n)$ time.

Time complexity of `parent`(j). Examining the listing (Figure 1), one observes that the number of operations is constant. However, though implicit in the listing, one should take into account the update of the data structures `child`(j) as well as the data structure that stores the set K, each taking $O(\log n)$ time.

Time complexity of `next-child`. First consider the time complexity of a single invocation of `next-child`, involving the update of the various data structures: The call to `parent` takes $O(\log n)$ time. Therefore, until the start of the repeat loop, the time is $O(\log n)$ (the time complexity of `succ` is $O(\log n)$). Now, the time complexity of a single iteration of the loop (without taking into account recursive calls to `next-child`) is $(O \log^2 n)$ because:

- The call to `toss` takes O(1) time.
- Finding the vertex of rank h in $[a, n+1] \setminus K$ takes $O(\log^2 n)$ time.
- Each of the $O(1)$ updates of $front(\cdot)$ or $front^{-1}(\cdot)$ may change the set K, and therefore may take $O(\log n)$ time to update the data structure involving K.
- Each update of a pointer $u(\cdot)$ results also in an (implicit) update in a certain `child` search tree, taking $O(\log n)$ time.

We now examine the number of iterations of the loop.

▶ **Claim 9.** *With high probability, the number of iterations of the loop in a single invocation of `next-child` is $O(\log n)$.*

Proof. We consider a process where the iterations continue until the selected node is node b. A random variable, R, depicting this number dominates a random variable that depicts the actual number of iterations. For each iteration, an additional node is selected by `toss`. By Lemma 7 the probability that a node $j < b$ is selected by `toss` is $1/\varphi(j)$, and we have that $1/\varphi(j) \leq \frac{1}{j-1}$. Thus, $R = 1 + \sum_{j=a}^{b-1} X_j$, where X_j is 1 iff node j was selected, 0 otherwise. Since $\mu = \sum_{j=a}^{b-1} \frac{1}{\varphi(j)} \leq \log n$, using Chernoff bound we have, for any constant $c > 6$, $P[R > c \cdot \log n] \leq 2^{-c \cdot \log n} = n^{-\Omega(1)}$. ◀

We thus have the following.

▶ **Lemma 10.** *For any given invocation of `next-child`, with high probability, the time complexity is $O(\log^3 n)$.*

5.2.3 Randomness complexity

In procedure `parent` we use $O(\log n)$ random bits whenever, for a given j, this procedure is called with parameter j for the first time.

In procedure `toss` the `if` condition holds with probability $1 - 1/n^{c-1}$ (where c is the constant used in that procedure). Therefore, given an invocation of `toss`, with probability $1 - 1/n^{c-1}$ this procedure uses $O(\log n)$ bits. By Claim 9, in each invocation of `next-child` the number of times that `toss` is called is, w.h.p., $O(\log n)$. We thus have the following.

▶ **Lemma 11.** *During a given call to* `next-child`*, w.h.p.,* $O(\log^2 n)$ *random bits are used.*

The following lemma states the time, space, and randomness complexities of the queries.

▶ **Lemma 12.** *The complexities of* `next-child-tp` *and* `parent` *are as follows.*

- *Given an invocation of* `parent` *the following hold for this invocation:*
  1. *The increase, during that invocation, of the space used by our algorithm is* $O(1)$*.*
  2. *The number of random bits used during that invocation is* $O(\log n)$*.*
  3. *The time complexity of that invocation is* $O(\log n)$*.*
- *Given an invocation of* `next-child-tp`*, with high probability, all of the following hold for this invocation:*
  1. *The increase, during that invocation, of the space used by our algorithm is* $O(\log^2 n)$*.*
  2. *The number of random bits used during that invocation is* $O(\log^4 n)$*.*
  3. *The time complexity of that invocation is* $O(\log^5 n)$*.*

Proof.

`parent`. During an invocation of `parent`(j) the size of the used space increases when a pointer $u(j)$ becomes non-nul or when additional values are stored in `child`$(u(j))$. To select $u(j)$, $O(\log n)$ random bits are used, and $O(\log n)$ time is consumed to insert j in `child`$(u(j))$ and to update the data structure for the set K (this is implicit in the listing).

`next-child-tp`. We first consider `next-child`. Observe that by Lemma 8, w.h.p., each and every root (non-recursive) invocation of `next-child` has a recursion tree of size $O(\log n)$. In each invocation of `next-child`, $O(1)$ variables *front*(j) and $u(j)$ may be updated. Therefore, w.h.p., for all root (non-recursive) calls to `next-child` it holds that the increase in space during this invocation is $O(\log n)$ (see Section 5.2.1). Using Lemmas 11 and 8 we have that, w.h.p., each root invocation of `next-child` uses $O(\log^3 n)$ random bits. Using Lemmas 10 and 8, we have that, w.h.p., the time complexity of each root invocation of `next-child` is $O(\log^4 n)$.

Because the types of the pointers are uniformly distributed in {`dir`, `rec`}, each call to `next-child-tp` results, w.h.p., in $O(\log n)$ calls to `next-child`. The above complexities are thus multiplied by an $O(\log n)$ factor to get the (w.h.p.) complexities of `next-child-tp`. ◀

6 On-the-fly Generator for BA-Graphs

Our on-the-fly generator for BA-graphs is called `O-t-F-BA`, and simply calls `BA-next-neighbor`(v) for each query on node v. We present an implementation for the `BA-next-neighbor` query, and prove its correctness, as well as analyze its time, space, and randomness complexities. The on-the-fly BA generator maintains n standard `heaps`, one for each node. The heaps store nodes, where the order is the natural order of their serial numbers. The heap of node j stores some of the nodes already known to be neighbors of j.

- *For the first* `BA-next-neighbor`(v) *query*, for a given v, we proceed as follows. We find the parent of v in the BA-graph, which is done by following, in the pointers tree, the pointers of the ancestors of v until we find an ancestor pointed to by a `dir` pointer (and not a `rec` pointer). See Figure 2. In addition, we initialize the process of finding neighbors of v to its right (i.e., with a bigger serial number) by inserting into the heap of v the "final node" $n+1$ as well as the first child of v.
- Observe that any *subsequent* `BA-next-neighbor`(v) *query* is to return a *child* of v in the BA-graph. The children x of v in the BA-graph have, in the pointers tree, a path of $u(\cdot)$ pointers starting at x and ending at v with all pointers, except the last one, being `rec`

```
1: procedure BA-NEXT-NEIGHBOR(v)
2:     if first_query(v) = true then
3: /* first query for v */
4:         first_query(v) ← false
5:         heap-insert(heap_v, n + 1)
6:         heap-insert(heap_v, next-child-tp(v, v, dir))
7:         return BA-parent(v)
8:     else
9: /* all subsequent queries for v */
10:        r ← heap-extract-min(heap_v)
11:        if r = n + 1 then
12:            heap-insert(heap_v, n + 1)
13:            return n + 1
14:        else
15:            if type(r) = dir then
16:                heap-insert(heap_v, next-child-tp(v, r, dir))
17:                heap-insert(heap_v, next-child-tp(r, r, rec))
18:            else
19:                (q, type) ← parent(r)
20:                heap-insert(heap_v, next-child-tp(q, r, rec))
21:            end if
22:            return r
23:        end if
24:    end if
25: end procedure
```

```
1: procedure BA-PARENT(v)
2:     (i, flag) ← parent(v)
3:     if flag = dir then
4:         return i
5:     else
6:         return BA-parent(i)
7:     end if
8: end procedure
```

Figure 2 Pseudo code of the on-the-fly BA generator.

(the last being `dir`). The query has to report the children in increasing index number. To this end the `heap` of v is used; it stores some of the children of v, *not yet returned by a* `BA-next-neighbor`(v) *query*. This heap is also updated so that `BA-next-neighbor`(v) will continue to return the next child according to the index order. To do so, whenever a node, r, is extracted from the heap, the heap is updated to include the following:

- If r has a `dir` pointer to v, then we add to the heap (1) the next, after r, node with a `dir` pointer to v, and (2) the first node that has a `rec` pointer to r.
- If r has a `rec` pointer to a node r', then we add to the heap the first, after r, node with a `rec` pointer to r'.

The proof of the next lemma, by induction on the number of queries, is omitted.

▶ **Lemma 13.** *The procedure `BA-next-neighbor` returns the next neighbor of v.*

Since the flags in the pointers tree are uniformly distributed, and by Lemma 12, we have:

▶ **Lemma 14.** *For any given root (non-recursive) invocation of `BA-parent`, with high probability, that invocation takes $O(\log^2 n)$ time.*

The next theorem follows from the code, standard heap implementation, and Lemma 12.

▶ **Theorem 15.** *For any given invocation of `BA-next-neighbor`, with high probability, all of the following hold for that invocation:*

1. *The increase, during that invocation, of the space used by our algorithm $O(\log^2 n)$.*
2. *The number of random bits used during that invocation is $O(\log^4 n)$.*
3. *The time complexity of that invocation is $O(\log^5 n)$.*

We now state the properties of our on-the-fly graph generator for BA-graphs.

▶ **Definition 16.** For a number of queries $T > 0$ and a sequence of `BA-next-neighbor` queries $Q = (q_1, \ldots, q_T)$, let $A(Q)$ be the sequence of answers returned by an algorithm A on Q. If A is randomized then $A(Q)$ is a probability distribution on sequences of answers.

Let $\texttt{Opt-BA}_n$ be the (randomized) algorithm that first runs the Markov process to generate a graph G on n nodes according to the BA model, stores G, and then answers queries by accessing the stored G. Let $\texttt{O-t-F-BA}_n$ be the algorithm `O-t-F-BA` run with graph-size n.

▶ **Theorem 17.** *For any sequence of queries Q, $\texttt{Opt-BA}_n(Q) = \texttt{O-t-F-BA}_n(Q)$.*

▶ **Theorem 18.** *For any $T > 0$ and any sequence of queries $Q = (q_1, \ldots, q_T)$, when using $\texttt{O-t-F-BA}_n$ it holds w.h.p. that, for all $1 \leq t \leq T$:*

1. *The increase in the used space, while processing query t, is $O(\log^2 n)$.*
2. *The number of random bits used while processing query t is $O(\log^4 n)$.*
3. *The time complexity for processing query t is $O(\log^5 n)$.*

Proof. A query $\texttt{BA-next-neighbor}(v)$ at time t is a *trivial* if at some $t' < t$ a query $\texttt{BA-next-neighbor}(v)$ returns $n+1$. Observe that trivial queries take $O(\log n)$ *deterministic* time, do not use randomness, and do not increase the used space. Since there are less than n^2 *non-trivial* queries, the theorem follows from Theorem 15 and a union bound. ◀

Acknowledgments. We thank Yishay Mansour for raising the question of whether one can locally generate preferential attachment graphs, and Dimitri Achlioptas and Matya Katz for useful discussions. We further thank an anonymous ICALP reviewer for a comment that helped us simplify one of the data structure implementations.

References

1 Md. Maksudul Alam, Maleq Khan, and Madhav V. Marathe. Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC'13, Denver, CO, USA – November 17-21, 2013*, pages 91:1–91:12, 2013. doi:10.1145/2503210.2503291.

2 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1132–1139, 2012. URL: http://dl.acm.org/citation.cfm?id=2095205.

3 Albert-László Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi:10.1126/science.286.5439.509.

4 Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3):036113, 2005.

5 Michael Drmota. *Random Trees: An Interplay Between Combinatorics and Probability*. Springer Publishing Company, Incorporated, 1st edition, 2009.

6 Guy Even, Moti Medina, and Dana Ron. Best of two local models: Local centralized and local distributed algorithms. *CoRR*, abs/1402.3796, 2014. URL: http://arxiv.org/abs/1402.3796.

7 Guy Even, Moti Medina, and Dana Ron. Deterministic stateless centralized local algorithms for bounded degree graphs. In *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 394–405, 2014. doi:10.1007/978-3-662-44777-2_33.

8 Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM J. Comput.*, 39(7):2761–2822, 2010. doi:10.1137/080722771.

9 Tamara G. Kolda, Ali Pinar, Todd Plantenga, and C. Seshadhri. A scalable generative graph model with community structure. *SIAM J. Scientific Computing*, 36(5), 2014. doi:10.1137/130914218.

10 Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D. Sivakumar, Andrew Tomkins, and Eli Upfal. Random graph models for the web graph. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 57–65, 2000. doi:10.1109/SFCS.2000.892065.

11 Reut Levi, Guy Moshkovitz, Dana Ron, Ronitt Rubinfeld, and Asaf Shapira. Constructing near spanning trees with few local inspections. *CoRR*, abs/1502.00413, 2015. URL: http://arxiv.org/abs/1502.00413.

12 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local algorithms for sparse spanning graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 826–842, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.826.

13 Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Brief announcement: Local computation algorithms for graphs of non-constant degrees. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 59–61, 2015. doi:10.1145/2755573.2755615.

14 Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 653–664, 2012. doi:10.1007/978-3-642-31594-7_55.

15 Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, pages 260–273, 2013. doi:10.1007/978-3-642-40328-6_19.

16 Ulrich Meyer and Manuel Penschuck. Generating massive scale-free networks under resource constraints. In *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016*, pages 39–52, 2016. doi:10.1137/1.9781611974317.4.

17 Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 327–336. IEEE, 2008.

18 Sadegh Nobari, Xuesong Lu, Panagiotis Karras, and Stéphane Bressan. Fast random graph generation. In *Proceedings of the 14th international conference on extending database technology*, pages 331–342. ACM, 2011.

19 Krzysztof Onak. New sublinear methods in the struggle against classical problems. *Massachusetts Institute of Technology, PhD Thesis*, September 2010.

20 Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1123–1131, 2012. URL: http://dl.acm.org/citation.cfm?id=2095204.

21 Omer Reingold and Shai Vardi. New techniques and tighter bounds for local computation algorithms. *J. Comput. Syst. Sci.*, 82(7):1180–1200, 2016. doi:10.1016/j.jcss.2016.05.007.

22 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Innovations in Computer Science – ICS 2010, Tsinghua University, Beijing, China,*

January 7-9, 2011. Proceedings, pages 223–238, 2011. URL: http://conference.itcs.tsinghua.edu.cn/ICS2011/content/papers/36.html.

23 Martin Sauerhoff. On the entropy of models for the web graph. Manuscript. URL: http://ls2-www.cs.uni-dortmund.de/~sauerhof/papers/ent.pdf.

24 Robert T. Smythe and Hosam M. Mahmoud. A survey of recursive trees. *Theory of Probability and Mathematical Statistics*, (51):1–28, 1995.

25 Andy Yoo and Keith W. Henderson. Parallel generation of massive scale-free graphs. *CoRR*, abs/1003.3684, 2010. URL: http://arxiv.org/abs/1003.3684.

26 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM J. Comput.*, 41(4):1074–1093, 2012. doi:10.1137/110828691.

Sublinear Time Estimation of Degree Distribution Moments: The Degeneracy Connection*

Talya Eden[†1], **Dana Ron**[‡2], **and C. Seshadhri**[3]

1 **School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel**
talyaa01@gmail.com

2 **School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel**
danaron@tau.ac.il

3 **Department of Computer Science, University of California – Santa Cruz, Santa Cruz, CA, USA**
sesh@ucsc.edu

Abstract

We revisit the classic problem of estimating the degree distribution moments of an undirected graph. Consider an undirected graph $G = (V, E)$ with n (non-isolated) vertices, and define (for $s > 0$) $\mu_s = \frac{1}{n} \cdot \sum_{v \in V} d_v^s$. Our aim is to estimate μ_s within a multiplicative error of $(1+\varepsilon)$ (for a given approximation parameter $\varepsilon > 0$) in sublinear time. We consider the sparse graph model that allows access to: uniform random vertices, queries for the degree of any vertex, and queries for a neighbor of any vertex. For the case of $s = 1$ (the average degree), $\widetilde{O}(\sqrt{n})$ queries suffice for any constant ε (Feige, SICOMP 06 and Goldreich-Ron, RSA 08). Gonen-Ron-Shavitt (SIDMA 11) extended this result to all integral $s > 0$, by designing an algorithms that performs $\widetilde{O}(n^{1-1/(s+1)})$ queries. (Strictly speaking, their algorithm approximates the number of star-subgraphs of a given size, but a slight modification gives an algorithm for moments.)

We design a new, significantly simpler algorithm for this problem. In the worst-case, it exactly matches the bounds of Gonen-Ron-Shavitt, and has a much simpler proof. More importantly, the running time of this algorithm is connected to the *degeneracy* of G. This is (essentially) the maximum density of an induced subgraph. For the family of graphs with degeneracy at most α, it has a query complexity of $\widetilde{O}\left(\frac{n^{1-1/s}}{\mu_s^{1/s}}\left(\alpha^{1/s} + \min\{\alpha, \mu_s^{1/s}\}\right)\right) = \widetilde{O}(n^{1-1/s}\alpha/\mu_s^{1/s})$. Thus, for the class of bounded degeneracy graphs (which includes all minor closed families and preferential attachment graphs), we can estimate the average degree in $\widetilde{O}(1)$ queries, and can estimate the variance of the degree distribution in $\widetilde{O}(\sqrt{n})$ queries. This is a major improvement over the previous worst-case bounds. Our key insight is in designing an estimator for μ_s that has low variance when G does not have large dense subgraphs.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity, F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Sublinear algorithms, Degree distribution, Graph moments

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.7

* The full version of this extended abstract is available at https://arxiv.org/abs/1604.03661.

† This research was partially supported by a grant from the Blavatnik fund. The author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

‡ This research was partially supported by the Israel Science Foundation grant No. 671/13 and by a grant from the Blavatnik fund.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 7; pp. 7:1–7:13

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

1 Introduction

Estimating the mean and moments of a sequence of n integers $d_1, d_2, \ldots, d_n$ is a classic problem in statistics that requires little introduction. In the absence of any knowledge of the moments of the sequence, it is not possible to prove anything non-trivial. But suppose these integers formed the degree sequence of a graph. Formally, let $G = (V, E)$ be an undirected graph over n vertices, and let d_v denote the degree of vertex $v \in V$, where we assume that $d_v \geq 1$ for every v.[1] Feige proved that $O^*(\sqrt{n})$ uniform random vertex degrees (in expectation) suffice to provide a $(2+\varepsilon)$-approximation to the average degree [23]. (We use $O^*(\cdot)$ to suppress $\text{poly}(\log n, 1/\varepsilon)$ factors.) The variance can be as large as n for graphs of constant average degree (simply consider a star), but the constraints of a degree distribution allow for non-trivial approximations. Classic theorems of Erdős-Gallai and Havel-Hakimi characterize such sequences [29, 21, 27].

Again, the star graph shows that the $(2+\varepsilon)$-approximation cannot be beaten in sublinear time through pure vertex sampling. Suppose we could also access random neighbors of a given vertex. In this setting, Goldreich and Ron showed it is possible to obtain a $(1+\varepsilon)$-approximation to the average degree in $O^*(\sqrt{n})$ expected time [24].

In a substantial (and complex) generalization, Gonen, Ron, and Shavitt (henceforth, GRS) gave a sublinear-time algorithm that estimates the higher moments of the degree distribution [25]. Technically, GRS gave an algorithm for approximating the number of stars in a graph, but a simple modification yields an algorithm for moments estimation. For precision, let us formally define this problem. The *degree distribution* is the distribution over the degree of a uniform random vertex. The *s-th moment* of the degree distribution is $\mu_s \triangleq \frac{1}{n} \cdot \sum_{v \in V} d_v^s$.

The Degree Distribution Moment Estimation (DDME) Problem. *Let $G = (V, E)$ be a graph over n vertices, where n is known. Access to G is provided through the following queries. We can (i) get the id (label) of a uniform random vertex, (ii) query the degree d_v of any vertex v, (iii) query a uniform random neighbor of any vertex v. Given $\varepsilon > 0$ and $s \geq 1$, output a $(1+\varepsilon)$-multiplicative approximation to μ_s with probability[2] $> 2/3$.*

The DDME problem has important connections to network science, which is the study of properties of real-world graphs. There have been numerous results on the significance of heavy-tailed/power-law degree distributions in such graphs, since the seminal results of Barabási-Albert [5, 10, 22]. The degree distribution and its moments are commonly used to characterize and model graphs appearing in varied applications [7, 36, 14, 37, 8]. On the theoretical side, recent results provide faster algorithms for graphs where the degree distribution has some specified form [6, 9]. Practical algorithms for specific cases of DDME have been studied by Dasgupta et al and Chierichetti et al. [17, 13]. (These results requires bounds on the mixing time of the random walk on G.)

1.1 Results

Let m denote the number of edges in the graph (where m is not provided to the algorithm). For the sake of simplicity, we restrict the discussion in the introduction to case when $\mu_s \leq n^{s-1}$.

[1] The assumption on there being no isolated vertices is made here only for the sake of simplicity of the presentation, as it ensures a basic lower bound on the moments.

[2] The constant $2/3$ is a matter of convenience. It can be increased to at least $1-\delta$ by taking the median value of $\log(1/\delta)$ independent invocations.

As observed by GRS, the complexity of the DDME problem is smaller when μ_s is significantly larger. GRS designed an (expected) $O^*\left(n^{1-1/(s+1)}/\mu_s^{1/(s+1)} + n^{1-1/s}\right)$-query algorithm for DDME and proved this expression was optimal up to $\text{poly}(\log n, 1/\varepsilon)$ dependencies. (Here $O^*(\cdot)$ also suppresses additional factors that depend only on s). Note that for a graph without isolated vertices, $\mu_s \geq 1$ for every $s > 0$, so this yields a worst-case $O^*(n^{1-1/(s+1)})$ bound. The $s = 1$ case is estimating the average degree, so this recovers the $O^*(\sqrt{n})$ bounds of Goldreich-Ron. We mention a recent result by Aliakbarpour et al. [1] for DDME, in a stronger model that assumes additional access to uniform random edges. They get a better bound of $O^*(m/(n\mu_s)^{1/s})$ in this stronger model, for $s > 1$ (and $\mu_s \leq n^{s-1}$). Note that the main challenge of DDME is in measuring the contribution of high-degree vertices, which becomes substantially easier when random edges are provided. In the DDME problem without such samples, it is quite non-trivial to even detect high degree vertices.

All the bounds given above are known to be optimal, up to $\text{poly}(\log n, 1/\varepsilon)$ dependencies, and at first blush, this problem appears to be solved. We unearth a connection between DDME and the *degeneracy* of G. The degeneracy of G is (up to a factor 2) the maximum density over all subgraphs of G. We design an algorithm that has a nuanced query complexity, depending on the degeneracy of G. Our result subsumes all existing results, and provides substantial improvements in many interesting cases. Furthermore, our algorithm and its analysis are significantly simpler and more concise than in the GRS result.

We begin with a convenient corollary of our main theorem. A tighter, more precise bound appears as Theorem 3.

▶ **Theorem 1.** *Consider the family of graphs with degeneracy at most α. The DDME problem can be solved on this family using $O^*\left(\frac{n^{1-1/s}}{\mu_s^{1/s}}\left(\alpha^{1/s} + \min\{\alpha, \mu_s^{1/s}\}\right)\right)$ queries in expectation. The running time is linear in the number of queries.*

Consider the case of *bounded degeneracy* graphs, where $\alpha = O(1)$. This is a rich class of graphs. *Every* minor-closed family of graphs has bounded degeneracy, as do graphs generated by the Barabási-Albert preferential attachment process [5]. There is a rich theory of *bounded expansion graphs*, which spans logic, graph minor theory, and fixed-parameter tractability [32]. All these graph classes have bounded degeneracy. For every such class of graphs, we get a $(1+\varepsilon)$-estimate of μ_s in $O^*(n^{1-1/s}/\mu_s^{1/s})$ time. We stress that bounded degeneracy does not imply any bounds on the maximum degree or the moments. The star graph has degeneracy 1, but has extremely large moments due to the central vertex.

Consider any bounded degeneracy graph without isolated vertices. We can accurately estimate the average degree ($s = 1$) in $\text{poly}(\log n)$ queries, and estimate the variance of the degree distribution ($s = 2$) in $\sqrt{n} \cdot \text{poly}(\log n)$ queries. Contrast this with the (worst-case optimal) $\sqrt{n}$ bounds of Feige and Goldreich-Ron for average degree, and the $O^*(n^{2/3})$ bound of GRS for variance estimation. For general s, our bound is a significant improvement over the $O^*(n^{1-1/(s+1)}/\mu_s^{1/(s+1)})$ bound of GRS.

The algorithm attaining Theorem 1 requires an upper bound on the degeneracy of the graph. When an degeneracy bound is not given, the algorithm recovers the bounds of GRS, with an improvement on the extra $\text{poly}(\log n)/\varepsilon$ factors. More details are in Theorem 3. We note that the degeneracy-dependent bound in Theorem 1 cannot be attained by an algorithm that is only given n as a parameter. In particular, if an algorithm is only provided with n and must work on all graphs with n vertices, then it must perform $\Omega(\sqrt{n})$ queries in order to approximate the average degree even for graphs of constant degeneracy (and constant average degree). Details are given in Subsection 7.1 in the full version of the paper.

The bound of Theorem 1 may appear artificial, but we prove that it is optimal when $\mu_s \leq n^{s-1}$. (For the general case, we also have optimal upper and lower bounds.) This construction is an extension of the lower bound proof of GRS.

▶ **Theorem 2.** *Consider the family of graphs with degeneracy α and where $\mu_s \leq n^{s-1}$. Any algorithm for the DDME problem on this family requires $\Omega\Big(\frac{n^{1-1/s}}{\mu_s^{1/s}} \cdot \Big(\alpha^{1/s} + \min\{\alpha, \mu_s^{1/s}\}\Big)\Big)$ queries.*

1.2 From degeneracy to moment estimation

We begin with a closer look at the lower bound examples of Feige, Goldreich-Ron, and GRS. The core idea is quite simple: DDME is hard when the overall graph is sparse, but there are small dense subgraphs. Consider the case of a clique of size $100\sqrt{n}$ connected to a tree of size n. The small clique dominates the average degree, but any sublinear algorithm with access only to random vertices pays $\Omega(\sqrt{n})$ for a non-trivial approximation. GRS use more complex constructions to get an $\Omega(n^{1-1/(s+1)})$ lower bound for general s. This also involves embedding small dense subgraphs that dominate the moments.

Can we prove a converse to these lower bound constructions? In other words, prove that the *non-existence* of dense subgraphs must imply that DDME is easier? A convenient parameter for this non-existence is the *degeneracy*.

But the degeneracy is a global parameter, and it is not clear how a sublinear algorithm can exploit it. Furthermore, DDME algorithms are typically very local; they sample random vertices, query the degrees of these vertices and maybe also query the degrees of some of their neighbors. We need a local property that sublinear algorithms can exploit, but can also be linked to the degeneracy. We achieve this connection via the *degree ordering* of G. Consider the DAG obtained by directing all edges from lower to higher degree vertices. Chiba-Nishizeki related the properties of the *out-degree* distribution to the degeneracy, and exploited this for clique counting [12]. Nonetheless, there is no clear link to DDME. (Nor do we use any of their techniques; we state this result merely to show what led us to use the degree ordering).

Our main insight is the construction of an estimator for DDME whose variance depends on the degeneracy of G. This estimator critically uses the degree ordering. Our proof relates the variance of this estimator to the density of subgraphs in G, which can be bounded by the degeneracy. We stress that our algorithm is quite simple, and the technicalities are in the analysis and setting of certain parameters.

1.3 Designing the algorithm

Designate the *weight* of an edge (u, v) to be $d_u^{s-1} + d_v^{s-1}$. A simple calculation yields that the sum of the weights of all edges is exactly $M_s \triangleq \sum_v d_v^s = n \cdot \mu_s$. Suppose we could sample uniform random *edges* (and knew the total number of edges). Then we could hope to estimate M_s through uniform edge sampling. The variance of the edge weights can be bounded, and this yields an $O^*(m/(n\mu_s)^{1/s}) = O^*(n^{1-1/s})$ algorithm (when no vertex is isolated). Indeed, this is very similar to the approach of Aliakbarpour et al. [1]. Such variance calculations were also used in the classic Alon-Matias-Szegedy result of frequency moment estimation [3].

Our approach is to simulate uniform edge samples using uniform vertex samples. Suppose we sampled a set R of uniform random vertices. By querying the degrees of all these vertices, we can select vertices in R with probability proportional to their degrees, which allows us to uniformly sample edges that are incident to vertices in R. Now, we simply run the uniform

edge sampling algorithm on these edges. This algorithmic structure was recently used for sublinear triangle counting algorithms by Eden et al. [19].

Here lies the core technical challenge. How to bound the number of random vertices that is sufficient for effectively simulating the random edge algorithm? This boils down to the behavior of the variance of the "vertex weight" distribution. Let the weight of a vertex be the sum of weights of its incident edges. The weight distribution over vertices can be extremely skewed, and this approach would require a forbiddingly large R.

A standard technique from triangle counting (first introduced by Chiba-Nishizeki [12]) helps reduce the variance. Direct all edges from lower degree to higher degree vertices, breaking ties consistently. Now, set the weight of a vertex to be the sum of weights on incident *out-edges*. Thus, a high-degree vertex with lower degree neighbors will have a significantly reduced weight, reducing overall variance. In the general case (ignoring degeneracy), a relatively simple argument bounds the maximum weight of a vertex, which enables us to bound the variance of the weight distribution. This yields a much simpler algorithm and proof of the GRS bound.

In the case of graphs with bounded degeneracy, we need a more refined approach. Our key insight is an intimate connection between the variance and the existence of dense subgraphs in G. We basically show that the main structure that leads to high variance is the existence of dense subgraphs. Formally, we can translate a small upper bound on the density of any subgraph to a bound on the variance of the vertex weights. This establishes the connection to the graph degeneracy.

1.4 Simplicity of our algorithm

Our viewpoint on DDME is quite different from GRS and its precursor [24], which proceed by bucketing the vertices based on their degree. This leads to a complicated algorithm, which essentially samples to estimate the size of the buckets, and also the number of edges between various buckets (and "sub-buckets"). We make use of buckets in out analysis, in order to obtain the upper bound that depends on the degeneracy α (in order to achieve the GRS upper bound, our analysis does not use bucketing).

As explained above, our main DDME procedure, **Moment-estimator** is simple enough to present in a few lines of pseudocode (see Figure 1). We feel that the structural simplicity of **Moment-estimator** is an important contribution of our work.

Moment-estimator takes two sampling parameters r and q. The main result Theorem 3 follows from running **Moment-estimator** with a standard geometric search for the right setting of r and q. In **Moment-estimator** we use $id(v)$ to denote the label of a vertex v, where vertices have unique ids and there is a complete order over the ids.

1.5 Other related work

As mentioned at the beginning of this section, Aliakbarpour et al. [1] consider the problem of approximating the number of s-stars for $s \geq 2$ when given access to uniformly selected edges. Given the ability to uniformly select edges, they can select vertices with probability proportional to their degree (rather than uniformly). This can be used to get an unbiased estimator of μ_s (or the s-star count) with low variance. This leads to an $O(m/(n\mu_s)^{1/s})$ bound, which is optimal (for $\mu_s \leq n^{s-1}$).

Dasgupta, Kumar, and Sarlos give practical algorithms for average degree estimation, though they assume bounds on the mixing time of the random walk on the graph [17]. A recent paper of Chierichetti et al. build on these methods to sample nodes according to

Moment-estimator$_s(r, q)$

1. Select r vertices, uniformly, independently, at random and let the resulting multi-set be denoted by R. Query the degree of each vertex in R, and let $d_R = \sum_{v \in R} d_v$.
2. For $i = 1, \ldots, q$ do:
   a. Select a vertex v_i with probability proportional to its degree (i.e., with probability d_{v_i}/d_R), and query for a random neighbor u_i of v_i.
   b. If $d_{v_i} < d_{u_i}$ or $d_{v_i} = d_{u_i}$ and $id(v_i) < id(u_i)$, set $X_i = (d_{v_i}^{s-1} + d_{u_i}^{s-1})$. Else, set $X_i = 0$.
3. Return $X = \frac{1}{r} \cdot \frac{d_R}{q} \cdot \sum_{i=1}^{q} X_i$.

Figure 1 Algorithm **Moment-estimator$_s$** for approximating μ_s.

powers of their degree (which is closely related to DDME) [13]. Simpson, Seshadhri, and McGregor give practical algorithms to estimate the entire cumulative degree distribution in the streaming setting [38]. This is different from the sublinear query model we consider, and the results are mostly empirical.

In [19], Eden et al. present an algorithm for approximating the number of triangles in a graph. Although this is a very different problem than DDME, there are similar challenges regarding high-degree vertices. Indeed, as mentioned earlier, the approach of sampling random edges through a set of random vertices was used in [19].

The degeneracy is closely related to other "density" notions, such as the arboricity, thickness, and strength of a graph [4]. There is a rich history of algorithmic results where run time depends on the degeneracy [31, 12, 2, 20].

Other sublinear algorithms for estimating various graph parameters include: approximating the size of the minimum-weight spanning tree [11, 16, 15], maximum matching [33, 39] and of the minimum vertex cover [35, 33, 30, 39, 28, 34].

A Comment regarding this extended abstract

We defer some of the details of the analysis of the algorithm, as well as the lower bound proof, to the accompanying full version of the paper.

2 The main theorem

▶ **Theorem 3.** *For every graph G, there exists an algorithm that returns a value Z such that $Z \in [(1-\varepsilon)\mu_s(G), (1+\varepsilon)\mu_s(G)]$ with probability at least $2/3$. Assume that algorithm is given α, an upper bound on the degeneracy of G. (If no such bound is provided, the algorithm assumes a trivial bound of $\alpha = \infty$.) The expected running time is the minimum of the following two expressions.*

$$O\left(2^s \cdot n^{1-1/s} \cdot \log^2 n \cdot \left(\frac{\alpha}{\mu_s}\right)^{1/s} + \min\left\{\frac{n^{1-1/s} \cdot \alpha}{\mu_s^{1/s}}, \frac{n^{s-1} \cdot \alpha}{\mu_s}\right\}\right) \cdot \frac{s \log n \cdot \log(s \log n)}{\varepsilon^2} \quad (1)$$

$$O\left(\frac{n^{1-1/(s+1)}}{\mu_s^{1/(s+1)}} + \min\left\{n^{1-1/s}, \frac{n^{s-1-1/s}}{\mu_s^{1-1/s}}\right\}\right) \cdot \frac{s \log n \cdot \log(s \log n)}{\varepsilon^2} \quad (2)$$

Equation (2) is essentially the query complexity of GRS (albeit with a better dependence on $s, \log n$, and $1/\varepsilon$). Thus, our algorithm is guaranteed to be at least as good as that. If α is

exactly the degeneracy of G, then we can prove that Equation (1) is less than Equation (2). Within each expression, there is a min of two terms. The first term is smaller iff $\mu_s \leq n^{s-1}$.

The mechanism of deriving this rather cumbersome running time is the following. The algorithm of Theorem 3 runs **Moment-estimator** for geometrically increasing values of r and q, which is in turn derived from a geometrically decreasing guess of μ_s. It uses this guess to set r and q. There is a setting of values depending on α, and a setting independent of it. The algorithm simply picks the minimum of these settings to achieve the smaller running time.

3 Sufficient conditions for r and q in Moment-estimator

In this section we provide sufficient conditions on the parameters r and q that are used by **Moment-estimator** (Figure 1), in order for the algorithm to return a $(1+\varepsilon)$ estimate of μ_s. First we introduce some notations. For a graph $G = (V, E)$ and a vertex $v \in V$, let $\Gamma(v)$ denote the set of neighbors of v in G (so that $d_v = |\Gamma(v)|$). For any (multi-)set R of vertices, let E_R be the (multi-)set of edges incident to the vertices in R. We will think of the edges in E_R as ordered pairs; thus (v, u) is distinct from (u, v), and so $E_R \triangleq \{(v, u) : v \in R, u \in \Gamma(v)\}$. Observe that d_R, as defined in Step 1 of **Moment-estimator** equals $|E_R|$. Let $M_s = M_s(G) \triangleq \sum_{v \in V} d_v^s$, so that $\mu_s = M_s/n$. In the analysis of the algorithm, it is convenient to work with M_s instead of μ_s.

A critical aspect of our algorithm (and proof) is the *degree ordering on vertices*. Formally, we set $u \prec v$ if $d_u < d_v$ or, $d_u = d_v$ and $id(u) < id(v)$. Given the degree ordering, we let $\Gamma^+(v) \triangleq \{u \in \Gamma(v) : v \prec u\}$, $d_v^+ \triangleq |\Gamma^+(v)|$, and $E^+ \triangleq \{(v, u) : v \in V, u \in \Gamma^+(v)\}$. Here and elsewhere, we use $\sum_v$ as a shorthand for $\sum_{v \in V}$.

▶ **Definition 4.** We define the weight of an edge $e = (v, u)$ as follows: if $v \prec u$ define $\mathrm{wt}(e) \triangleq (d_v^{s-1} + d_u^{s-1})$. Otherwise, $\mathrm{wt}(e) \triangleq 0$.
For a vertex $v \in V$, $\mathrm{wt}(v) \triangleq \sum_{u \in \Gamma(v)} \mathrm{wt}((v, u)) = \sum_{u \in \Gamma^+(v)} \mathrm{wt}((v, u))$, and for a (multi-)set of vertices R, $\mathrm{wt}(R) \triangleq \sum_{v \in R} \mathrm{wt}(v)$.

Observe that given the above notations and definition, **Moment-estimator** selects uniform edges from E_R and sets each X_i (in Step 2b) to $\mathrm{wt}((v_i, u_i))$. The next two claims readily follow from Definition 4 (and the description of the algorithm).

▶ **Claim 5.** $\sum_v \mathrm{wt}(v) = M_s$.

▶ **Claim 6.** $\mathrm{Ex}[X] = \mu_s$, *where* X *is as defined in Step 3 of the algorithm.*

3.1 Conditions on the parameters r and q

We next state two conditions on the parameters r and q, which are used in the algorithm, and then establish several claims, based on the conditions holding. The conditions are stated in terms of properties of the graph as well as the approximation parameter ε and a confidence parameter δ.

1. **The vertex condition:** $r \geq (120 \cdot n \cdot \sum_v \mathrm{wt}(v)^2)/(\varepsilon^2 \cdot \delta \cdot M_s^2)$,
2. **The edge condition:** $q \geq 2000 \cdot m \cdot M_{2s-1}/(\varepsilon^2 \cdot \delta^3 \cdot M_s^2)$.

▶ **Lemma 7.** *If Condition 1 holds, then with probability at least* $1 - \delta/2$*, all the following hold.*

1. $\mathrm{wt}(R) \in \left[\left(1 - \frac{\varepsilon}{2}\right) \cdot \frac{r}{n} \cdot M_s, \left(1 + \frac{\varepsilon}{2}\right) \cdot \frac{r}{n} \cdot M_s\right]$.
2. $|E_R| \leq \frac{12}{\delta} \cdot \frac{r}{n} \cdot m$.
3. $\sum_{(v,u) \in E_R^+} \mathrm{wt}\left((v,u)\right)^2 \leq \frac{18}{\delta} \cdot \frac{r}{n} \cdot M_{2s-1}$.

The proof of the first item in Lemma 7 follows from Chebyshev's inequality (using $\mathrm{Var}[\mathrm{wt}(R)] \leq \frac{r}{n} \cdot \sum_v \mathrm{wt}(v)^2$), and the proofs of the other two items follow from Markov's inequality (as well as the definition of M_{2s-1}).

▶ **Theorem 8.** *If Conditions 1 and 2 hold, then* $X \in [(1-\varepsilon)\mu_s, (1+\varepsilon)\mu_s]$ *with probability at least* $1 - \delta$*.*

Proof. Condition on any choice of R. We have $\mathrm{Ex}[X|R] = (1/r)\mathrm{wt}(R)$. Turning to the variance, since the edges (v_i, u_i) are chosen from E_R uniformly at random, it is not hard to verify that

$$\mathrm{Var}[X|R] = \left(\frac{1}{r}\right)^2 \cdot \left(\frac{|E_R|}{q}\right)^2 \cdot \mathrm{Var}\left[\sum_{i=1}^{q} X_i \;\middle|\; R\right] = \frac{1}{q} \cdot \frac{|E_R|}{r} \cdot \frac{\sum_{(v,u) \in E_R^+} \mathrm{wt}\left((v,u)\right)^2}{r} \,.$$

Let us now condition on R such that the bounds of Lemma 7 hold. Note that such an R is chosen with probability at least $1 - \delta/2$. We get $\mathrm{Var}[X|R] \leq \frac{250}{\delta^2} \cdot \frac{1}{q} \cdot \frac{m}{n} \cdot \frac{M_{2s-1}}{n}$. We apply Chebyshev's inequality and invoke Condition 2:

$$\Pr\left[\left|(X|R) - \mathrm{Ex}[X|R]\right| \leq \frac{\varepsilon}{2} \cdot \mu_s\right] \;\leq\; \frac{4 \cdot \mathrm{Var}[X|R]}{\varepsilon^2 \cdot \mu_s^2} \leq \frac{1}{q} \cdot \frac{4 \cdot (250/\delta^2) \cdot m \cdot M_{2s-1}}{\varepsilon^2 \cdot M_s^2} \leq \frac{\delta}{2} \,.$$

By Lemma 7, $\mathrm{Ex}[X|R] = (1/r)\mathrm{wt}(R) \in [(1-\varepsilon/2)\mu_s, (1+\varepsilon/2)\mu_s]$. The theorem follows by applying the union bound. ◀

4 Satisfying Conditions 1 and 2 in general graphs

We show how to set r and q to satisfy Conditions 1 and 2 in general graphs. Our setting of r and q will give us the same query complexity as [25] (up to the dependence on $1/\varepsilon$ and $\log n$, on which we improve, and the exponential dependence on s in [25], which we do not incur). In the next section we show how the setting of r and q can be improved using a degeneracy bound.

For c_r and c_q that are sufficiently large constants, we set

$$r = \frac{c_r}{\varepsilon^2 \cdot \delta} \cdot \frac{n}{M_s^{1/(s+1)}} \,, \qquad q = \frac{c_q}{\varepsilon^2 \cdot \delta^3} \cdot \min\left\{n^{1-1/s}, \frac{n^{s-1/s}}{M_s^{1-1/s}}\right\} \,. \tag{3}$$

This setting of parameters requires the knowledge of M_s, which is exactly what we are trying to approximate (up to the normalization factor of n). A simple geometric search argument alleviates the need to know M_s. For details see Section 6.

In order to assert that r as set in Equation (3) satisfies Condition 1, it suffices to establish the next lemma.

▶ **Lemma 9** (Condition 1 holds)**.** $\sum_v \mathrm{wt}(v)^2 \leq 4M_s^{2-\frac{1}{s+1}}$.

Proof. Let $\theta = M_s^{1/(s+1)}$ be a degree threshold. We define $H \triangleq \{v : d_v > \theta\}$, $L \triangleq V \setminus H$. This partition into "high-degree" vertices (H) and "low-degree" vertices (L) will be useful in upper bounding the maximum weight $\mathrm{wt}(v)$ of a vertex v, and hence upper bounding $\sum_v \mathrm{wt}(v)^2$. Details follow.

We first observe that $|H| \leq M_s^{1/(s+1)}$. This is true since otherwise, $\sum_{v\in H} d_v^s > M_s^{1/(s+1)} \cdot M_s^{\frac{s}{s+1}} = M_s$, which is a contradiction. We claim that this upper bound on $|H|$ implies that

$$\max_v d_v^+ \leq M_s^{1/(s+1)} \ . \tag{4}$$

To verify this, assume, contrary of the claim, that for some v, $d_v^+ > M_s^{1/(s+1)}$. But then there are at least $M_s^{1/(s+1)}$ vertices u such that $d_u \geq d_v \geq d_v^+ > M_s^{1/(s+1)}$. This contradicts the bound on $|H|$.

It will also be useful to bound $\sum_{u\in H} d_u^{s-1}$. By Hölder's inequality with conjugates s and $s/(s-1)$ (a statement of Hölder's inequality can be found in the full version of the paper) and the bound on $|H|$,

$$\sum_{u\in H} d_u^{s-1} = \sum_{u\in H} 1 \cdot d_u^{s-1} \leq |H|^{1/s} \left(\sum_{u\in H} d_u^s\right)^{\frac{s-1}{s}} \leq M_s^{\frac{1}{s(s+1)}} \cdot M_s^{\frac{s-1}{s}} \leq M_s^{\frac{s}{s+1}} \ . \tag{5}$$

We now turn to bounding $\max_v\{\mathrm{wt}(v)\}$. By the definition of $\mathrm{wt}(v)$ and the degree ordering,

$$\mathrm{wt}(v) = \sum_{u\in\Gamma^+(v)} (d_v^{s-1} + d_u^{s-1}) \leq 2 \sum_{u\in\Gamma^+(v)} d_u^{s-1} = 2 \sum_{u\in\Gamma^+(v)\cap L} d_u^{s-1} + 2 \sum_{u\in\Gamma^+(v)\cap H} d_u^{s-1} \ . \tag{6}$$

For the first term on the right-hand-side of Equation (6), recall that $d_u \leq M_s^{1/(s+1)}$ for $u \in L$. Thus, by Equation (4),

$$\sum_{u\in\Gamma^+(v)\cap L} d_u^{s-1} \leq d_v^+ \cdot M_s^{\frac{s-1}{s+1}} \leq M_s^{\frac{s}{s+1}} \ . \tag{7}$$

For the second term, using $\Gamma^+(v) \cap H \subseteq H$ and applying Equation (5),

$$\sum_{u\in\Gamma^+(v)\cap H} d_u^{s-1} \leq \sum_{u\in H} d_u^{s-1} \leq M_s^{\frac{s}{s+1}} \ . \tag{8}$$

Finally,

$$\sum_v \mathrm{wt}(v)^2 \leq \max_v\{\mathrm{wt}(v)\} \cdot \sum_v \mathrm{wt}(v) \leq M_s^{2-1/(s+1)} \ ,$$

where the second inequality follows by combining Equations (6)–(8) to get an upper bound on $\max_v\{\mathrm{wt}(v)\}$ and applying Claim 5. ◀

The next lemma implies that Condition 2 holds for q as set in Equation (3).

▶ **Lemma 10** (Condition 2 holds). $\min\left\{n^{1-1/s}, \frac{n^{s-1/s}}{M_s^{1-1/s}}\right\} \geq 2m \cdot \frac{M_{2s-1}}{M_s^2}$.

Proof. We can bound M_{2s-1} in two ways. First, by a standard norm inequality, since $s \geq 1$,

$$M_{2s-1} = \sum_v d_v^{2s-1} \leq \left(\sum_v d_v^s\right)^{(2s-1)/s} = M_s^{2-1/s} \ . \tag{9}$$

We can also use the trivial bound $d_v \leq n$ and get $M_{2s-1} \leq n^{s-1} \cdot M_s$. Thus, $M_{2s-1} \leq \min\{M_s^{2-1/s}, n^{s-1} \cdot M_s\}$. By applying Hölder's inequality with conjugates $s/(s-1)$ and s we get that

$$2m = \sum_v 1 \cdot d_v \leq n^{(s-1)/s} \cdot \left(\sum_v d_v^s \right)^{1/s} = n^{1-1/s} \cdot M_s^{1/s} \,. \tag{10}$$

We multiply the bound by M_{2s-1} to complete the proof. ◀

5 The Degeneracy Connection

The degeneracy, or the coloring number, of a graph $G = (V, E)$ is the maximum value, over all subgraphs G' of G, of the minimum degree in G'. In this definition, we can replace "minimum" by "average" to get a 2-factor approximation to the degeneracy (refer to [26]; Theorem 2.4.4 and Corollary 5.2.3 of [18]). Abusing notation, it will be convenient for us to define $\alpha(G) = \max_{S \subseteq V} \left\{ \frac{|E(S)|}{|S|} \right\}$.

We also make the following observation regarding the relation between $\alpha(G)$ and $M_s(G)$.

▶ **Claim 11.** *For every graph G, $\alpha(G) \leq M_s(G)^{\frac{1}{s+1}}$.*

In this section, we show that the following setting of parameters for **Moment-estimator**$_s$ satisfies Conditions 1 and 2, for every graph G with degeneracy at most α (i.e., $\alpha(G) \leq \alpha$), and for appropriate constants c_r and c_q.

$$r = \frac{c_r}{\varepsilon^2 \cdot \delta} \cdot \min \left\{ \frac{n}{M_s^{1/(s+1)}} \,,\; 2^s \cdot n \cdot \log^2 n \cdot \left(\frac{\alpha}{M_s} \right)^{1/s} \right\} , \tag{11}$$

$$q = \frac{c_q}{\varepsilon^2 \cdot \delta^3} \cdot \min \left\{ \frac{n \cdot \alpha}{M_s^{1/s}}, \frac{n^s \cdot \alpha}{M_s}, n^{1-1/s}, \frac{n^{s-1/s}}{M_s^{1-1/s}} \right\} . \tag{12}$$

Clearly the setting of r and q in Equation (11) and Equation (12) respectively, can only improve on the setting of r and q for the general case in Equation (3) (Section 4).

Our main challenge is in proving that Condition 1 holds for r as set in Equation (11) (when the graph has degeneracy at most α). Here too, the goal is to upper bound $\sum_v \mathrm{wt}(v)^2$. However, as opposed to the proof of Lemma 9 in Section 4, where we simply obtained an upper bound on $\max_v\{\mathrm{wt}(v)\}$ (and bounded $\sum_v \mathrm{wt}(v)^2$ by $\max_v\{\mathrm{wt}(v)\} \cdot M_s$), here the analysis is more refined, and uses the degeneracy bound. For details see the proof of our main lemma, stated next.

▶ **Lemma 12** (Condition 1 holds). *For a sufficiently large constant c, $\sum_v \mathrm{wt}(v)^2 \leq c \cdot 2^s \cdot \alpha^{1/s} \cdot M_s^{2-1/s} \cdot \log^2 n$.*

Proof Sketch. In this extended abstract we only provide the high-level structure of the proof. By the definition of $\mathrm{wt}(v)$, and since $d_v \leq d_u$ for every v and $u \in \Gamma^+(v)$,

$$\sum_v \mathrm{wt}(v)^2 = \sum_v \Big(\sum_{u \in \Gamma^+(v)} \left(d_v^{s-1} + d_u^{s-1} \right) \Big)^2 \;\leq\; 4 \cdot \sum_v \Big(\sum_{u \in \Gamma^+(v)} d_u^{s-1} \Big)^2 . \tag{13}$$

In order to bound the expression on the right-hand-side of Equation (13) we partition the vertices (with degree at least 1) according to their degree. Let $U_i \triangleq \{u \in V : d_u \in$

$(2^{i-1}, 2^i]\}$ for $0 \le i \le \lceil \log n \rceil$, and let $\Gamma_i^+(v)$ be a shorthand for $\Gamma^+(v) \cap U_i$. By considering each U_i separately and applying Hölder's inequality we get the following bound for every v.

$$\sum_{u \in \Gamma_i^+(v)} 1 \cdot d_u^{s-1} \le |\Gamma_i^+(v)|^{1/s} \cdot \Big(\sum_{u \in \Gamma_i^+(v)} d_u^s \Big)^{(s-1)/s} \le |\Gamma_i^+(v)|^{1/s} \cdot M_s^{(s-1)/s} \,. \tag{14}$$

For each i, we also partition the vertices in V according to the number of outgoing edges that they have to U_i. Specifically, for $1 \le j \le \lceil \log(n/\alpha) \rceil$, define $V_{i,j} \triangleq \Big\{ v \in V : |\Gamma_i^+(v)| \in \big(2^{j-1}\alpha, 2^j\alpha\big] \Big\}$. Also define $V_{i,0} \triangleq \Big\{ v \in V : |\Gamma_i^+(v)| \le \alpha \Big\}$. Hence, $\{V_{i,j}\}_{j=0}^{\lceil \log(n/\alpha) \rceil}$ is a partition of V for each i.

For a vertex u, let $\Gamma^-(u) \triangleq \{v : u \in \Gamma^+(v)\}$. For two sets of vertices S and T (which are not necessarily disjoint), let $E^+(S,T) \triangleq \{(u,v) : (u,v) \in E^+, u \in S, v \in T\}$. By applying Equation (14) (to one term of the square $\Big(\sum_{u \in \Gamma_i^+(v)} d_u^{s-1}\Big)^2$), and by the definition of $V_{i,j}$, it can be shown that

$$\sum_v \Big(\sum_{u \in \Gamma_i^+(v)} d_u^{s-1} \Big)^2 \le M_s^{(s-1)/s} \cdot \sum_{j=0}^{\lceil \log n \rceil} \Big(\sum_{u \in U_i} d_u^{s-1} \cdot \sum_{v \in \Gamma^-(u) \cap V_{i,j}} |\Gamma_i^+(v)|^{1/s} \Big) \,. \tag{15}$$

For $j < 2$ we can show that $\sum_{u \in U_i} d_u^{s-1} \sum_{v \in \Gamma^-(u) \cap V_{i,j}} |\Gamma_i^+(v)|^{1/s} \le 2 \cdot \alpha^{1/s} \cdot M_s$. Turning to $j \ge 2$, since all vertices in U_i have degree at most 2^i, we get:

$$\sum_{u \in U_i} d_u^{s-1} \cdot \sum_{v \in \Gamma^-(u) \cap V_{i,j}} |\Gamma_i^+(v)|^{1/s} \le 2^{j/s} \cdot \alpha^{1/s} \cdot 2^{i(s-1)} \cdot |E^+(V_{i,j}, U_i)| \,. \tag{16}$$

Since G has degeneracy at most α and by the definition of $V_{i,j}$, it can be shown that $|E^+(V_{i,j}, U_i)| \le 2\alpha \cdot |U_i|$, where $U_i = U_i \cap \Big(\bigcup_{v \in V_{i,j}} \Gamma^+(v) \Big)$. Furthermore, the definition of U_i (together with the degeneracy bound and the definition of M_s) implies that $|U_i| \le M_s \cdot 2^{-((i-1)(s-1)+j)} \cdot \alpha^{-1}$. The lemma follows by combining Equation (13) with Equation (15) and the above bounds for $j < 2$ and $j \ge 2$. ◀

The next lemma, which establishes Condition 2, can be proved similarly to Lemma 10.

▶ **Lemma 13** (Condition 2 holds).

$$\min \left\{ \frac{n \cdot \alpha}{M_s^{1/s}}, \frac{n^s \cdot \alpha}{M_s}, n^{1-1/s}, \frac{n^{s-1/s}}{M_s^{1-1/s}} \right\} \ge m \cdot \frac{M_{2s-1}}{M_s^2} \,.$$

6 Wrapping things up

The proof of our final result, Theorem 3, follows by combining Theorem 8, Lemma 9, Lemma 12 and Lemma 13, with a geometric search for a factor-2 estimate of M_s (which determines the correct setting of r and q in the algorithm).

References

1 A. S. Aliakbarpour, M.and Biswas, T. Gouleakis, J. Peebles, R. Rubinfeld, and A. Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, pages 1–30, 2017. doi:10.1007/s00453-017-0287-3.

2 N. Alon and S. Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. In *Proceedings of the Annual International Conference Computing and Combinatorics (COCOON)*, pages 394–405, 2008.

3 N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

4 Arboricity. Wikipedia. `https://en.wikipedia.org/wiki/Arboricity`.

5 A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, October 1999.

6 J. W. Berry, L. A. Fostvedt, D. J. Nordman, C. A. Phillips, C. Seshadhri, and A. G. Wilson. Why do simple algorithms for triangle enumeration work in the real world? *Internet Mathematics*, 11(6):555–571, 2015.

7 Z. Bi, C. Faloutsos, and F. Korn. The dgx distribution for mining massive, skewed data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 17–26. ACM, 2001.

8 P. Bickel, A. Chen, and E. Levina. The method of moments and degree distributions for network models. *Annals of Statistics*, 39(5):2280–2301, 2011.

9 P. Brach, M. Cygan, J. Lacki, and P. Sankowski. Algorithmic complexity of power law networks. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 1306–1325, 2016.

10 A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33:309–320, 2000.

11 B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing*, 34(6):1370–1379, 2005.

12 N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14:210–223, 1985.

13 F. Chierichetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlos. On sampling nodes in a network. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 471–481, 2016.

14 A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

15 A. Czumaj, F. Ergun, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005.

16 A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009.

17 A. Dasgupta, R. Kumar, and T. Sarlos. On estimating the average degree. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 795–806, 2014.

18 R. Diestel. *Graph Theory*. Springer, fourth edition edition, 2010.

19 T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 614–633, 2015.

20 D. Eppstein, M. Loffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 403–413, 2010.

21 P. Erdos and T. Gallai. Graphs with prescribed degree of vertices (hungarian). *Mat. Lapok*, 11:264–274, 1960.

22 M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of Computer Communication Review (SIGCOMM)*, pages 251–262. ACM, 1999.

23 U. Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.

24 O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Structures and Algorithms*, 32(4):473–493, 2008.

25 M. Gonen, D. Ron, and Y. Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Math*, 25(3):1365–1411, 2011.

26 Graph degeneracy. Wikipedia. `https://en.wikipedia.org/wiki/Degeneracy_(graph_theory)`.

27 S. L. Hakimi. On the realizability of a set of integers as degrees of the vertices of a graph. *SIAM Journal Applied Mathematics*, 10:496–506, 1962.

28 A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 22–31. IEEE, 2009.

29 V. Havel. A remark on the existence of finite graphs (czech). *Casopis Pest. Mat.*, 80:477–480, 1955.

30 S. Marko and D. Ron. Approximating the distance to properties in bounded-degree and general sparse graphs. *ACM Transactions on Algorithms*, 5(2), 2009.

31 D. Matula and L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983.

32 J. Nešetřil and P. Ossana de Mendez. *Sparsity*. Springer, 2010.

33 H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 327–336. IEEE, 2008.

34 K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 1123–1131. SIAM, 2012.

35 M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.

36 D. Pennock, G. Flake, S. Lawrence, E. Glover, and C. L. Giles. Winners don't take all: Characterizing the competition for links on the web. *Proceedings of the national academy of sciences (PNAS)*, 99(8):5207–5211, 2002.

37 A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao. Measurement-calibrated graph models for social network experiments. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 861–870. ACM, 2010.

38 O. Simpson, C. Seshadhri, and A. McGregor. Catching the head, tail, and everything in between: A streaming algorithm for the degree distribution. In *Proceedings on the International Conference on Data Mining (ICDM)*, pages 979–984, 2015.

39 Y. Yoshida, M. Yamamoto, and H. Ito. An improved constant-time approximation algorithm for maximum-matchings. In *Proceedings of the Annual Symposium on the Theory of Computing (STOC)*, pages 225–234. ACM, 2009.

Near-Optimal Closeness Testing of Discrete Histogram Distributions*†

Ilias Diakonikolas[1], Daniel M. Kane[2], and Vladimir Nikishkin[3]

1 **University of Southern California, Los Angeles, CA, USA**
diakonik@usc.edu
2 **University of California, San Diego, CA, USA**
dakane@cs.ucsd.edu
2 **University of Edinburgh, Edinburgh, UK**
v.nikishkin@sms.ed.ac.uk

Abstract

We investigate the problem of testing the equivalence between two discrete histograms. A k-*histogram* over $[n]$ is a probability distribution that is piecewise constant over some set of k intervals over $[n]$. Histograms have been extensively studied in computer science and statistics. Given a set of samples from two k-histogram distributions p, q over $[n]$, we want to distinguish (with high probability) between the cases that $p = q$ and $\|p - q\|_1 \geq \epsilon$. The main contribution of this paper is a new algorithm for this testing problem and a nearly matching information-theoretic lower bound. Specifically, the sample complexity of our algorithm matches our lower bound up to a logarithmic factor, improving on previous work by polynomial factors in the relevant parameters. Our algorithmic approach applies in a more general setting and yields improved sample upper bounds for testing closeness of other structured distributions as well.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.3 Probability and Statistics

Keywords and phrases distribution testing, histograms, closeness testing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.8

1 Introduction

In this work, we study the problem of testing equivalence (closeness) between two discrete *structured* distributions. Let $\mathcal{D}$ be a family of univariate distributions over $[n]$ (or $\mathbb{Z}$). The problem of *closeness testing for* $\mathcal{D}$ is the following: Given sample access to two unknown distribution $p, q \in \mathcal{D}$, we want to distinguish between the case that $p = q$ versus $\|p - q\|_1 \geq \epsilon$. (Here, $\|p-q\|_1$ denotes the ℓ_1-distance between the distributions p, q.) The sample complexity of this problem depends on the underlying family $\mathcal{D}$.

For example, if $\mathcal{D}$ is the class of *all* distributions over $[n]$, then it is known [13] that the optimal sample complexity is $\Theta(\max\{n^{2/3}/\epsilon^{4/3}, n^{1/2}/\epsilon^2\})$. This sample bound is best possible only if the family $\mathcal{D}$ includes all possible distributions over $[n]$, and we may be able to obtain significantly better upper bounds for most natural settings. For example, if both p, q are promised to be (approximately) log-concave over $[n]$, there is an algorithm to test

* A full version of this paper is available at https://arxiv.org/abs/1703.01913.
† I. D. was supported by NSF Award CCF-1652862 (CAREER) and a Sloan Research Fellowship. D. K. was supported by NSF Award CCF-1553288 (CAREER) and a Sloan Research Fellowship. V. N. was supported by a University of Edinburgh PCD Scholarship.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 8; pp. 8:1–8:15


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

equivalence between them using $O(1/\epsilon^{9/4})$ samples [25]. This sample bound is independent of the support size n, and is dramatically better than the worst-case tight bound [13] when n is large.

More generally, [25] described a framework to obtain sample-efficient equivalence testers for various families of structured distributions over both continuous and discrete domains. While the results of [25] are sample-optimal for *some* families of distributions (in particular, over continuous domains), it was not known whether they can be improved for natural families of discrete distributions. In this paper, we work in the framework of [25] and obtain new nearly-matching algorithms and lower bounds.

Before we state our results in full generality, we describe in detail a concrete application of our techniques to the case of *histograms* – a well-studied family of structured discrete distributions with a plethora of applications.

Testing Closeness of Histograms. A *k-histogram* over $[n]$ is a probability distribution $p : [n] \to [0, 1]$ that is piecewise constant over some set of k intervals over $[n]$. The algorithmic difficulty in testing properties of such distributions lies in the fact that the location and "size" of these intervals is a priori unknown. Histograms have been extensively studied in statistics and computer science. In the database community, histograms [37, 14, 46, 33, 35, 36, 1] constitute the most common tool for the succinct approximation of data. In statistics, many methods have been proposed to estimate histogram distributions [44, 32, 45, 40, 21, 48, 38] in a variety of settings.

In recent years, histogram distributions have attracted renewed interested from the theoretical computer science community in the context of learning [18, 10, 11, 12, 23, 2, 3, 27] and testing [36, 17, 26, 8, 9]. Here we study the following testing problem: Given sample access to two distributions p, q over $[n]$ that are promised to be (approximately) k-histograms, distinguish between the cases that $p = q$ versus $\|p - q\|_1 \geq \epsilon$. As the main application of our techniques, we give a new testing algorithm and a nearly-matching information-theoretic lower bound for this problem.

We now provide a summary of previous work on this problem followed by a description of our new upper and lower bounds. We want to ϵ-test closeness in ℓ_1-distance between two k-histograms over $[n]$, where $k \leq n$. Our goal is to understand the optimal sample complexity of this problem as a function of $k, n, 1/\epsilon$. Previous work is summarized as follows:

- In [25], the authors gave a closeness tester with sample complexity $O(\max\{k^{4/5}/\epsilon^{6/5}, k^{1/2}/\epsilon^2\})$.
- The best known sample lower bound is $\Omega(\max\{k^{2/3}/\epsilon^{4/3}, k^{1/2}/\epsilon^2\})$. This straightforwardly follows from [13], since k-histograms can simulate *any* support k distribution.

Notably, none of the two bounds depends on the domain size n. Observe that the upper bound of $O(\max\{k^{4/5}/\epsilon^{6/5}, k^{1/2}/\epsilon^2\})$ cannot be tight for the entire range of parameters. For example, for $n = O(k)$, the algorithm of [13] for testing closeness between arbitrary support n distributions has sample size $O(\max\{k^{2/3}/\epsilon^{4/3}, k^{1/2}/\epsilon^2\})$, matching the above sample complexity lower bound, up to a constant factor.

This simple example might suggest that the $\Omega(\max\{k^{2/3}/\epsilon^{4/3}, k^{1/2}/\epsilon^2\})$ lower bound is tight in general. We prove that this is not the case. The main conceptual message of our new upper bound and nearly-matching lower bound is the following:

> *The sample complexity of ϵ-testing closeness between two k-histograms over $[n]$ depends in a subtle way on the relation between the relevant parameters k, n and $1/\epsilon$.*

We find this fact rather surprising because such a phenomenon does *not* occur for the sample complexities of closely related problems. Specifically, testing the identity of a k-histogram

over $[n]$ to a *fixed* distribution has sample complexity $\Theta(k^{1/2}/\epsilon^2)$ [26]; and learning a k-histogram over $[n]$ has sample complexity $\Theta(k/\epsilon^2)$ [11]. Note that both these sample bounds are independent of n and are known to be tight for the entire range of parameters $k, n, 1/\epsilon$.

Our main positive result is a new closeness testing algorithm for k-histograms over $[n]$ with sample complexity $O\big(k^{2/3} \cdot \log^{4/3}(2 + n/k) \log(k)/\epsilon^{4/3}\big)$. Combined with the known upper bound of [25], we obtain the sample upper bound of

$$O\big(\max\big(\min(k^{4/5}/\epsilon^{6/5}, k^{2/3}\log^{4/3}(2+n/k)\log(k)/\epsilon^{4/3}), k^{1/2}\log^2(k)\log\log(k)/\epsilon^2\big)\big).$$

As our main negative result, we prove a lower bound of $\Omega(\min(k^{2/3}\log^{1/3}(2+n/k)/\epsilon^{4/3}, k^{4/5}/\epsilon^{6/5}))$. The first term in this expression shows that the "$\log(2 + n/k)$" factor that appears in the sample complexity of our upper bound is in fact necessary, up to a constant power. In summary, these bounds provide a nearly-tight characterization of the sample complexity of our histogram testing problem for the entire range of parameters.

A few observations are in order to interpret the above bounds:

- When n goes to infinity, the $O(k^{4/5}/\epsilon^{6/5})$ upper bound of [25] is tight for k-histograms.
- When $n = \mathrm{poly}(k)$ and ϵ is not too small (so that the $k^{1/2}/\epsilon^2$ term does not kick in), then the right answer for the sample complexity of our problem is $(k^{2/3}/\epsilon^{4/3})\mathrm{polylog}(k)$.
- The terms "$k^{4/5}/\epsilon^{6/5}$" and "$k^{2/3}\log^{4/3}(2 + n/k)\log(k)/\epsilon^{4/3}$" appearing in the sample complexity become equal when n is exponential in k. Therefore, our new algorithm has better sample complexity than that of [25] for all $n \leq 2^{O(k)}$.

In the following subsection, we state our results in a general setting and explain how the aforementioned applications are obtained from them.

1.1 Our Results and Comparison to Prior Work

For a given family $\mathcal{D}$ of discrete distributions over $[n]$, we are interested in designing a closeness tester for distributions in $\mathcal{D}$. We work in the general framework introduced by [26, 25]. Instead of designing a different tester for any given family $\mathcal{D}$, the approach of [26, 25] proceeds by designing a generic equivalence tester under a *different metric* than the ℓ_1-distance. This metric, termed $\mathcal{A}_k$-distance [20], where $k \geq 2$ is a positive integer, interpolates between Kolmogorov distance (when $k = 2$) and the ℓ_1-distance (when $k = n$). It turns out that, for a range of structured distribution families $\mathcal{D}$, the $\mathcal{A}_k$-distance can be used as a proxy for the ℓ_1-distance for a value of $k \ll n$ [11]. For example, if $\mathcal{D}$ is the family of k-histograms over $[n]$, the $\mathcal{A}_{2k}$ distance between them is tantamount to their ℓ_1 distance. We can thus obtain an ℓ_1 closeness tester for $\mathcal{D}$ by plugging in the right value of k in a general $\mathcal{A}_k$ closeness tester.

To formally state our results, we will need some terminology.

Notation. We will use p, q to denote the probability mass functions of our distributions. If p is discrete over support $[n] := \{1, \ldots, n\}$, we denote by p_i the probability of element i in the distribution. For two discrete distributions p, q, their ℓ_1 and ℓ_2 distances are $\|p-q\|_1 = \sum_{i=1}^n |p_i - q_i|$ and $\|p-q\|_2 = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$. Fix a partition of the domain I into disjoint intervals $\mathcal{I} := (I_i)_{i=1}^{\ell}$. For such a partition $\mathcal{I}$, the *reduced distribution* $p_r^{\mathcal{I}}$ corresponding to p and $\mathcal{I}$ is the discrete distribution over $[\ell]$ that assigns the i-th "point" the mass that p assigns to the interval I_i; i.e., for $i \in [\ell]$, $p_r^{\mathcal{I}}(i) = p(I_i)$. Let $\mathfrak{J}_k$ be the collection of all partitions of the domain I into k intervals. For $p, q : I \to \mathbb{R}_+$ and $k \in \mathbb{Z}_+$, we define the $\mathcal{A}_k$-distance between p and q by $\|p-q\|_{\mathcal{A}_k} \stackrel{\mathrm{def}}{=} \max_{\mathcal{I}=(I_i)_{i=1}^k \in \mathfrak{J}_k} \sum_{i=1}^k |p(I_i) - q(I_i)| = \max_{\mathcal{I}\in\mathfrak{J}_k} \|p_r^{\mathcal{I}} - q_r^{\mathcal{I}}\|_1$.

In this context, [25] gave a closeness testing algorithm under the $\mathcal{A}_k$-distance using $O(\max\{k^{4/5}/\epsilon^{6/5}, k^{1/2}/\epsilon^2\})$ samples. It was also shown that this sample bound is information–theoretically optimal (up to constant factors) for some adversarially constructed continuous distributions, or discrete distributions of support size n sufficiently large as a function of k. These results raised two natural questions: (1) What is the *optimal* sample complexity of the $\mathcal{A}_k$-closeness testing problem as a function of $n, k, 1/\epsilon$? (2) Can we obtain tight sample lower bounds for *natural* families of structured distributions?

We resolve both these open questions. Our main algorithmic result is the following:

▶ **Theorem 1.** *Given sample access to distributions p and q on $[n]$ and $\epsilon > 0$ there exists an algorithm that takes*

$$O\left(\max\left(\min\left(k^{4/5}/\epsilon^{6/5}, k^{2/3}\log^{4/3}(2+n/k)\log(2+k)/\epsilon^{4/3}\right), k^{1/2}\log^2(k)\log\log(k)/\epsilon^2\right)\right)$$

samples from each of p and q and distinguishes with $2/3$ probability between the cases that $p = q$ and $\|p-q\|_{\mathcal{A}_k} \geq \epsilon$.

As explained in [26, 25], using Theorem 1 one can obtain testing algorithms for the ℓ_1 closeness testing of various distribution families $\mathcal{D}$, by using the $\mathcal{A}_k$ distance as a "proxy" for the ℓ_1 distance:

▶ **Fact 2.** *For a univariate distribution family $\mathcal{D}$ and $\epsilon > 0$, let $k = k(\mathcal{D}, \epsilon)$ be the smallest integer such that for any $f_1, f_2 \in \mathcal{D}$ it holds that $\|f_1 - f_2\|_1 \leq \|f_1 - f_2\|_{\mathcal{A}_k} + \epsilon/2$. Then there exists an ℓ_1 closeness testing algorithm for $\mathcal{D}$ with the sample complexity of Theorem 1.*

Applications

Our upper bound for ℓ_1-testing of k-histogram distributions follows from the above by noting that for any k-histograms p, q we have $\|p-q\|_1 = \|p-q\|_{\mathcal{A}_{2k}}$. Also note that our upper bound is *robust*: it applies even if p, q are $O(\epsilon)$-close in ℓ_1-norm to being k-histograms.

Finally, we remark that our general $\mathcal{A}_k$ closeness tester yields improved upper bounds for various other families of structured distributions. Consider for example the case that $\mathcal{D}$ consists of all k-mixtures of some simple family (e.g., discrete Gaussians or log-concave), where the parameter k is large. The algorithm of [25] leads to a tester whose sample complexity scales with $O(k^{4/5})$, while Theorem 1 implies a $\tilde{O}(k^{2/3})$ bound.

On the lower bound side, we show:

▶ **Theorem 3.** *Let p and q be distributions on $[n]$ and let $\epsilon > 0$ be less than a sufficiently small constant. Any tester that distinguishes between $p = q$ and $\|p-q\|_{\mathcal{A}_k} \geq \epsilon$ for some $k \leq n$ must use $\Omega(m)$ samples for $m = \min(k^{2/3}\log^{4/3}(2+n/k)/\epsilon^{4/3}, k^{4/5}/\epsilon^{6/5})$.*

Furthermore, for $m = \min(k^{2/3}\log^{1/3}(2+n/k)/\epsilon^{4/3}, k^{4/5}/\epsilon^{6/5})$, any tester that distinguishes between $p = q$ and $\|p-q\|_{\mathcal{A}_k} \geq \epsilon$ must use $\Omega(m)$ samples even if p and q are both guaranteed to be piecewise constant distributions on $O(k+m)$ pieces.

Note that a lower bound of $\Omega(\sqrt{k}/\epsilon^2)$ straightforwardly applies even for p and q being k-histograms. This dominates the above bounds for $\epsilon < k^{-3/8}$.

We also note that our general lower bound with respect to the $\mathcal{A}_k$ distance is somewhat stronger, matching the term "$\log^{4/3}(2+n/k)$" in our upper bound.

1.2 Related Work

During the past two decades, *distribution property testing* [5] – whose roots lie in statistical hypothesis testing [41, 39] – has received considerable attention by the computer science community, see [43, 7] for two recent surveys. The majority of the early work in this field has focused on characterizing the sample size needed to test properties of arbitrary distributions of a given support size. After two decades of study, this "worst-case" regime is well-understood: for many properties of interest there exist sample-optimal testers (matched by information-theoretic lower bounds) [42, 13, 47, 26, 24, 22].

In many settings of interest, we know a priori that the underlying distributions have some "nice structure" (exactly or approximately). The problem of *learning* a probability distribution under such structural assumptions is a classical topic in statistics, see [4] for a classical book, and [34] for a recent book on the topic, that has recently attracted the interest of computer scientists [18, 19, 10, 16, 11, 12, 1, 30, 31, 28, 15, 3, 27, 29].

On the other hand, the theory of *distribution testing* under structural assumptions is less fully developed. More than a decade ago, Batu, Kumar, and Rubinfeld [6] considered a specific instantiation of this question – testing the equivalence between two unknown discrete monotone distributions – and obtained a tester whose sample complexity is poly-logarithmic in the domain size. A recent sequence of works [17, 26, 25] developed a framework to leverage such structural assumptions and obtained more efficient testers for a number of natural settings. However, for several natural properties of interest there is still a substantial gap between known sample upper and lower bounds.

1.3 Overview of Techniques

To prove our upper bound, we use a technique of iteratively reducing the number of bins (domain elements). In particular, we show that if we merge bins together in consecutive pairs, this does not significantly affect the $\mathcal{A}_k$ distance between the distributions, unless a large fraction of the discrepancy between our distributions is supported on $O(k)$ bins near the boundaries in the optimal partition. In order to take advantage of this, we provide a novel identity tester that requires few samples to distinguish between the cases where $p = q$ and the case where p and q have a large ℓ_1 distance supported on only k of the bins. We are able to take advantage of the small support essentially because having a discrepancy supported on few bins implies that the ℓ_2 distance between the distributions must be reasonably large.

Our new lower bounds are somewhat more involved. We prove them by exhibiting explicit families of pairs of distributions, where in one case $p = q$ and in the other p and q have large $\mathcal{A}_k$ distance, but so that it is information-theoretically impossible to distinguish between these two families with a small number of samples. In both cases, p and q are explicit piecewise constant distributions with a small number of pieces. In both cases, our domain is partitioned into a small number of bins and the restrictions of the distributions to different bins are independent, making our analysis easier. In some bins we will have $p = q$ each with mass about $1/m$ (where m is the number of samples). These bins will serve the purpose of adding "noise" making harder to read the "signal" from the other bins. In the remaining bins, we will have either that $p = q$ being supported on some interval, or p and q will be supported on consecutive, non-overlapping intervals. If three samples are obtained from any one of these intervals, the order of the samples and the distributions that they come from will provide us with information about which family we came from. Unfortunately, since triple collisions are relatively uncommon, this will not be useful unless $m \gg \max(k^{4/5}/\epsilon^{6/5}, k^{1/2}/\epsilon^2)$. Bins from which we have one or zero samples will tell us nothing, but bins from which we have exactly two samples may provide information.

For these bins, it can be seen that we learn nothing from the ordering of the samples, but we may learn something from their spacing. In particular, in the case where p and q are supported on disjoint intervals, we would suspect that two samples very close to each other are far more likely to be taken from the same distribution rather than from opposite distributions. On the other hand, in order to properly interpret this information, we will need to know something about the scale of the distributions involved in order to know when two points should be considered to be "close". To overcome this difficulty, we will stretch each of our distributions by a random exponential amount. This will effectively conceal any information about the scales involved so long as the total support size of our distributions is exponentially large.

2 A Near-Optimal Closeness Tester over Discrete Domains

2.1 Warmup: A Simpler Algorithm

We start by giving a simpler algorithm establishing a basic version of Theorem 1 with slightly worse parameters:

▶ **Proposition 4.** *Given sample access to distributions p and q on $[n]$ and $\epsilon > 0$ there exists an algorithm that takes*

$$O\left(k^{2/3}\log^{4/3}(3+n/k)\log\log(3+n/k)/\epsilon^{4/3} + \sqrt{k}\log^2(3+n/k)\log\log(3+n/k)/\epsilon^2\right)$$

samples from each of p and q and distinguishes with $2/3$ probability between the cases that $p = q$ and $\|p-q\|_{\mathcal{A}_k} \geq \epsilon$.

The basic idea of our algorithm is the following: From the distributions p and q construct new distributions p' and q' by merging pairs of consecutive buckets. Note that p' and q' each have much smaller domains (of size about $n/2$). Furthermore, note that the $\mathcal{A}_k$ distance between p and q is $\sum_{I\in\mathcal{I}}|p(I)-q(I)|$ for some partition $\mathcal{I}$ into k intervals. By using essentially the same partition, we can show that $\|p'-q'\|_{\mathcal{A}_k}$ should be almost as large as $\|p-q\|_{\mathcal{A}_k}$. This will in fact hold unless much of the error between p and q is supported at points near the endpoints of intervals in $\mathcal{I}$. If this is the case, it turns out there is an easy algorithm to detect this discrepancy. We require the following definitions:

▶ **Definition 5.** For a discrete distribution p on $[n]$, the merged distribution obtained from p is the distribution p' on $\lceil n/2\rceil$, so that $p'(i) \stackrel{\text{def}}{=} p(2i)+p(2i+1)$. For a partition $\mathcal{I}$ of $[n]$, define the *divided partition* $\mathcal{I}'$ of domain $\lceil n/2\rceil$, so that $I'_i \in \mathcal{I}'$ has the points obtained by point-wise gluing together odd points and even points.

Note that one can simulate a sample from p' given a sample from p by letting $p' = \lceil p/2\rceil$.

▶ **Definition 6.** Let p and q be distributions on $[n]$. For integers $k \geq 1$, let $\|p-q\|_{1,k}$ be the sum of the largest k values of $|p(i)-q(i)|$ over $i \in [n]$.

We begin by showing that either $\|p'-q'\|_{\mathcal{A}_k}$ is close to $\|p-q\|_{\mathcal{A}_k}$ or $\|p-q\|_{1,k}$ is large.

▶ **Lemma 7.** *For any two distributions p and q on $[n]$, let p' and q' be the merged distributions. Then,*

$$\|p-q\|_{\mathcal{A}_k} \leq \|p'-q'\|_{\mathcal{A}_k} + 2\|p-q\|_{1,k}\ .$$

Proof. Let $\mathcal{I}$ be the partition of $[n]$ into k intervals so that $\|p-q\|_{\mathcal{A}_k} = \sum_{I\in\mathcal{I}} |p(I)-q(I)|$. Let $\mathcal{I}'$ be obtained from $\mathcal{I}$ by rounding each upper endpoint of each interval except for the last down to the nearest even integer, and rounding the lower endpoint of each interval up to the nearest odd integer. Note that

$$\sum_{I\in\mathcal{I}'} |p(I)-q(I)| = \sum_{I\in\mathcal{I}'} |p'(I/2)-q'(I/2)| \le \|p'-q'\|_{\mathcal{A}_k} .$$

The partition $\mathcal{I}'$ is obtained from $\mathcal{I}$ by taking at most k points and moving them from one interval to another. Therefore, the difference

$$\left| \sum_{I\in\mathcal{I}} |p(I)-q(I)| - \sum_{I\in\mathcal{I}'} |p(I)-q(I)| \right| ,$$

is at most twice the sum of $|p(i)-q(i)|$ over these k points, and therefore at most $2\|p-q\|_{1,k}$. Combing this with the above gives our result. ◀

Next, we need to show that if two distributions have $\|p-q\|_{1,k}$ large that this can be detected easily.

▶ **Lemma 8.** *Let p and q be distributions on $[n]$. Let $k>0$ be a positive integer, and $\epsilon>0$. There exists an algorithm which takes $O(k^{2/3}/\epsilon^{4/3}+\sqrt{k}/\epsilon^2)$ samples from each of p and q and, with probability at least $2/3$, distinguishes between the cases that $p=q$ and $\|p-q\|_{1,k}>\epsilon$.*

Note that if we needed to distinguish between $p=q$ and $\|p-q\|_1>\epsilon$, this would require $\Omega(n^{2/3}/\epsilon^{4/3}+\sqrt{n}/\epsilon^2)$ samples. However, the optimal testers for this problem are morally ℓ_2-testers. That is, roughly, they actually distinguish between $p=q$ and $\|p-q\|_2>\epsilon/\sqrt{n}$. From this viewpoint, it is clear why it would be easier to test for discrepancies in $\|-\|_{1,k}$-distance, since if $\|p-q\|_{1,k}>\epsilon$, then $\|p-q\|_2>\epsilon/\sqrt{k}$, making it easier for our ℓ_2-type tester to detect the difference.

Our general approach will be by way of the techniques developed in [24]. We begin by giving the definition of a split distribution coming from that paper:

▶ **Definition 9.** Given a distribution p on $[n]$ and a multiset S of elements of $[n]$, define the *split distribution* p_S on $[n+|S|]$ as follows: For $1\le i\le n$, let a_i denote 1 plus the number of elements of S that are equal to i. Thus, $\sum_{i=1}^n a_i = n+|S|$. We can therefore associate the elements of $[n+|S|]$ to elements of the set $B=\{(i,j): i\in[n], 1\le j\le a_i\}$. We now define a distribution p_S with support B, by letting a random sample from p_S be given by (i,j), where i is drawn randomly from p and j is drawn randomly from $[a_i]$.

We now recall two basic facts about split distributions:

▶ **Fact 10** ([24])**.** *Let p and q be probability distributions on $[n]$, and S a given multiset of $[n]$. Then:*

(i) *We can simulate a sample from p_S or q_S by taking a single sample from p or q, respectively.*

(ii) *It holds $\|p_S-q_S\|_1 = \|p-q\|_1$.*

▶ **Lemma 11** ([24])**.** *Let p be a distribution on $[n]$. Then:*

(i) *For any multisets $S\subseteq S'$ of $[n]$, $\|p_{S'}\|_2 \le \|p_S\|_2$, and*

(ii) *If S is obtained by taking m samples from p, then $\mathbb{E}[\|p_S\|_2^2] \le 1/m$.*

We also recall an optimal ℓ_2 closeness tester under the promise that one of the distributions has smal ℓ_2 norm:

▶ **Lemma 12** ([13]). *Let p and q be two unknown distributions on $[n]$. There exists an algorithm that on input n, $b \geq \min\{\|p\|_2, \|q\|_2\}$ and $0 < \epsilon < \sqrt{2}b$, draws $O(b/\epsilon^2)$ samples from each of p and q and, with probability at least $2/3$, distinguishes between the cases that $p = q$ and $\|p - q\|_2 > \epsilon$.*

Proof of Lemma 8: We begin by presenting the algorithm:

Algorithm `Small-Support-Discrepancy-Tester`
Input: sample access to pdf's $p, q : [n] \to [0,1]$, $k \in \mathbb{Z}_+$, and $\epsilon > 0$.
Output: "YES" if $q = p$; "NO" if $\|q - p\|_{1,k} \geq \epsilon$.

1. Let $m = \min(k^{2/3}/\epsilon^{4/3}, k)$.
2. Let S be the multiset obtained by taking m independent samples from p.
3. Use the ℓ_2 tester of Lemma 12 to distinguish between the cases that $p_S = q_S$ and $\|p_S - q_S\|_2^2 \geq k^{-1}\epsilon^2/2$ and return the result.

The analysis is simple. By Lemma 11, with 90% probability $\|p_S\|_2 = O(m^{-1/2})$, and therefore the number of samples needed (using the ℓ_2 tester from Lemma 12) is $O(m + km^{-1/2}/\epsilon^2) = O(k^{2/3}/\epsilon^{4/3} + \sqrt{k}/\epsilon^2)$. If $p = q$, then $p_S = q_S$ and the algorithm will return "YES" with appropriate probability. If $\|q - p\|_{1,k} \geq \epsilon$, then $\|p_S - q_S\|_{1,k+m} \geq \epsilon$. Since $k + m$ elements contribute to total ℓ_1 error at least ϵ, by Cauchy-Schwarz, we have that $\|p_S - q_S\|_2^2 \geq \epsilon^2/(k+m) \geq k^{-1}\epsilon^2/2$. Therefore, in this case, the algorithm returns "NO" with appropriate probability. ◀

Proof of Proposition 4: The basic idea of our algorithm is the following: By Lemma 8, if $\|p - q\|_{\mathcal{A}_k}$ is large, then so is either $\|p - q\|_{1,k}$ or $\|p' - q'\|_{\mathcal{A}_k}$. Our algorithm then tests whether $\|p - q\|_{1,k}$ is large, and recursively tests whether $\|p' - q'\|_{\mathcal{A}_k}$ is large. Since p', q' have half the support size, we will only need to do this for $\log(n/k)$ rounds, losing only a poly-logarithmic factor in the sample complexity. We present the algorithm here:

Algorithm `Small-Domain-`$\mathcal{A}_k$`-tester`
Input: sample access to pdf's $p, q : [n] \to [0,1]$, $k \in \mathbb{Z}_+$, and $\epsilon > 0$.
Output: "YES" if $q = p$; "NO" if $\|q - p\|_{\mathcal{A}_k} \geq \epsilon$.

1. For $i := 0$ to $t \stackrel{\text{def}}{=} \lceil \log_2(n/k) \rceil$, let $p^{(i)}, q^{(i)}$ be distributions on $[\lceil 2^{-i} n \rceil]$ defined by $p^{(i)} = \lceil 2^{-i} p \rceil$ and $q^{(i)} = \lceil 2^{-i} q \rceil$.
2. Take $Ck^{2/3} \log^{4/3}(3 + n/k) \log\log(3 + n/k)/\epsilon^{4/3}$ samples, for C sufficiently large, and use these samples to distinguish between the cases $p^{(i)} = q^{(i)}$ and $\|p^{(i)} - q^{(i)}\|_{1,k} > \epsilon/(4\log_2(3+n/k))$ with probability of error at most $1/(10\log_2(3+n/k))$ for each i from 0 to t, using the same samples for each test.
3. If any test yields that $p^{(i)} \neq q^{(i)}$, return "NO". Otherwise, return "YES".

We now show correctness. In terms of sample complexity, we note that by taking a majority over $O(\log\log(3 + n/k))$ independent runs of the tester from Lemma 8 we can run this algorithm with the stated sample complexity. Taking a union bound, we can also assume that all tests performed in Step 2 returned the correct answer. If $p = q$ then $p^{(i)} = q^{(i)}$ for all i and thus, our algorithm returns "YES". Otherwise, we have that $\|p - q\|_{\mathcal{A}_k} \geq \epsilon$. By repeated application of Lemma 7, we have that

$$\|p - q\|_{\mathcal{A}_k} \leq \sum_{i=0}^{t-1} 2\|p^{(i)} - q^{(i)}\|_{1,k} + \|p^{(t)} - q^{(t)}\|_{\mathcal{A}_k} \leq 2\sum_{i=0}^{t} \|p^{(i)} - q^{(i)}\|_{1,k} \,,$$

where the last step was because $p^{(t)}$ and $q^{(t)}$ have a support of size at most k and so $\|p^{(t)} - q^{(t)}\|_{\mathcal{A}_k} = \|p^{(t)} - q^{(t)}\|_1 = \|p^{(t)} - q^{(t)}\|_{1,k}$. Therefore, if this is at least ϵ, it must be the case that $\|p^{(i)} - q^{(i)}\|_{1,k} > \epsilon/(4\log_2(3+n/k))$ for some $0 \leq i \leq t$, and thus our algorithm returns "NO". This completes our proof. ◀

2.2 Full Algorithm

The improvement to Proposition 4 is somewhat technical. The key idea involves looking into the analysis of Lemma 8. Generally speaking, choosing a larger value of m (up to the total sample complexity), will decrease the ℓ_2 norm of p, and thus the final complexity. Unfortunately, taking $m > k$ might lead to problems as it will subdivide the k original bins on which the error is supported into $\omega(k)$ bins. This in turn could worsen the lower bounds on $\|p - q\|_2$. However, this will only be the case if the total mass of these bins carrying the difference is large. Thus, we can obtain an improvement to Lemma 8 when the mass of bins on which the error is supported is small. The details are deferred to the full version.

3 Nearly Matching Information-Theoretic Lower Bound

We give a lower bound for k-histograms (k-flat distributions), postponing our slightly stronger construction to the full version. Before moving to the discrete setting, we first establish a lower bound for continuous histogram distributions. Our bound on discrete distributions will follow from taking the adversarial distribution from this example and rounding its values to the nearest integer. In order for this to work, we will need ensure to that our adversarial distribution does not have its $\mathcal{A}_k$-distance decrease by too much when we apply this operation. To satisfy this requirement, we will guarantee that our distributions will be piecewise constant with all the pieces of length at least 1.

▶ **Proposition 13.** *Let $k \in \mathbb{Z}_+$, $\epsilon > 0$ sufficiently small, and $W > 2$. Fix*

$$m = \min(k^{2/3}\log^{1/3}(W)/\epsilon^{4/3}, k^{4/5}/\epsilon^{6/5}) .$$

There exist distributions $\mathcal{D}, \mathcal{D}'$ over pairs of distributions p and q on $[0, 2(m+k)W]$, where p and q are $O(m+k)$-flat with pieces of length at least 1, so that: (a) when drawn from $\mathcal{D}$, we have $p = q$ deterministically, (b) when drawn from $\mathcal{D}'$, we have $\|p - q\|_{\mathcal{A}_k} > \epsilon$ with 90% probability, and so that $o(m)$ samples are insufficient to distinguish whether or not the pair is drawn from $\mathcal{D}$ or $\mathcal{D}'$ with better than 2/3 probability.

At a high-level, our lower bound construction proceeds as follows: We will divide our domain into $m+k$ bins so that no information about which distributions had samples drawn from a given bin or the ordering of these samples will help to distinguish between the cases of $p = q$ and otherwise, unless at least three samples are taken from the bin in question. Approximately k of these bins will each have mass ϵ/k and might convey this information if at least three samples are taken from the bin. However, the other m bins will each have mass approximately $1/m$ and will be used to add noise. In all, if we take s samples, we expect to see approximately $s^3\epsilon^3/k^2$ of the lighter bins with at least three samples. However, we will see approximately s^3/m^2 of our heavy bins with three samples. In order for the signal to overwhelm the noise, we will need to ensure that we have $(s^3\epsilon^3/k^2)^2 > s^3/m^2$.

The above intuitive sketch assumes that we cannot obtain information from the bins in which only two samples are drawn. This naively should not be the case. If $p = q$, the distance between two samples drawn from that bin will be independent of whether or not they are

drawn from the same distribution. However, if p and q are supported on disjoint intervals, one would expect that points that are close to each other should be far more likely to be drawn from the same distribution than from different distributions. In order to disguise this, we will scale the length of the intervals by a random, exponential amount, essentially making it impossible to determine what is meant by two points being close to each other. In effect, this will imply that two points drawn from the same bin will only reveal $O(1/\log(W))$ bits of information about whether $p = q$ or not. Thus, in order for this information to be sufficient, we will need that $(s^2\epsilon^2/k)^2/\log(W) > (s^2/m)$. We proceed with the formal proof below.

Proof of Proposition 13: We use ideas from [24] to obtain this lower bound using an information theoretic argument.

We may assume that $\epsilon > k^{1/2}$, because otherwise we may employ the standard lower bound that $\Omega(\sqrt{k}/\epsilon^2)$ samples are required to distinguish two distributions on a support of size k.

First, we note that it is sufficient to take $\mathcal{D}$ and $\mathcal{D}'$ be distributions over pairs of non-negative, piecewise constant distributions with total mass $\Theta(1)$ with 90% probability so that running a Poisson process with parameter $o(m)$ is insufficient to distinguish a pair from $\mathcal{D}$ from a pair from $\mathcal{D}'$ [24].

We construct these distributions as follows: We divide the domain into $m + k$ bins of length $2W$. For each bin i, we independently generate a random ℓ_i, so that $\log(\ell_i/2)$ is uniformly distributed over $[0, 2\log(W)/3]$. We then produce an interval I_i within bin i of total length ℓ_i and with random offset. In all cases, we will have p and q supported on the union of the I_i's.

For each i with probability $m/(m+k)$, we have the restrictions of p and q to I_i both uniform with $p(I_i) = q(I_i) = 1/m$. The other $k/(m + k)$ of the time we have $p(I_i) = q(I_i) = \epsilon/k$. In this latter case, if p and q are being drawn from $\mathcal{D}$, p and q are each constant on this interval. If they are being drawn from $\mathcal{D}'$, then $p + q$ will be constant on the interval, with all of that mass coming from p on a random half and coming from q on the other half.

Note that in all cases p and q are piecewise constant with $O(m + k)$ pieces of length at least 1. It is easy to show that with high probability the total mass of each of p and q is $\Theta(1)$, and that if drawn from $\mathcal{D}'$ that $\|p - q\|_{\mathcal{A}_k} \gg \epsilon$ with at least 90% probability.

We will now show that if one is given m samples from each of p and q, taken randomly from either $\mathcal{D}$ or $\mathcal{D}'$, that the shared information between the samples and the source family will be small. This implies that one is unable to consistently guess whether our pair was taken from $\mathcal{D}$ or $\mathcal{D}'$.

Let X be a random variable that is uniformly at random either 0 or 1. Let A be obtained by applying a Poisson process with parameter $s = o(m)$ on the pair of distributions p, q drawn from $\mathcal{D}$ if $X = 0$ or from $\mathcal{D}'$ if $X = 1$. We note that it suffices to show that the shared information $I(X : A) = o(1)$. In particular, by Fano's inequality, we have:

▶ **Lemma 14.** *If X is a uniform random bit and A is a correlated random variable, then if f is any function so that $f(A) = X$ with at least 51% probability, then $I(X : A) \geq 2 \cdot 10^{-4}$.*

Let A_i be the samples of A taken from the i^{th} bin. Note that the A_i are conditionally independent on X. Therefore, we have that $I(X : A) \leq \sum_i I(X : A_i) = (m + k)I(X : A_1)$. We will proceed to bound $I(X : A_1)$.

We note that $I(X : A_1)$ is at most the integral over pairs of multisets a (representing a set of samples from q and a set of samples from p), of

$$O\left(\frac{(\Pr(A_1 = a|X = 0) - \Pr(A_1 = a|X = 1))^2}{\Pr(A_1 = a)}\right).$$

Thus,

$$I(X : A_1) = \sum_{h=0}^{\infty} \int_{|a|=h} O\left(\frac{(\Pr(A_1 = a|X = 0) - \Pr(A_1 = a|X = 1))^2}{\Pr(A_1 = a)}\right).$$

We will split this sum up based on the value of h.

For $h = 0$, we note that the distributions for $p + q$ are the same for $X = 0$ and $X = 1$. Therefore, the probability of selecting no samples is the same. Therefore, this contributes 0 to the sum.

For $h = 1$, we note that the distributions for $p + q$ are the same in both cases, and conditioning on I_1 and $(p+q)(I_1)$ that $\mathbb{E}[p]$ and $\mathbb{E}[q]$ are the same in each of the cases $X = 0$ and $X = 1$. Therefore, again in this case, we have no contribution.

For $h \geq 3$, we note that $I(X : A_1) \leq I(X : A_1, I_1) \leq I(X : A_1|I_1)$, since I_1 is independent of X. We note that $\Pr(A_1 = a|X = 0, p(I_1) = 1/m) = \Pr(A_1 = a|X = 1, p(I_1) = 1/m)$. Therefore, we have that

$$\begin{aligned}\Pr(A_1 = a|X = 0) - \Pr(A_1 = a|X = 1)\\ = \Pr(A_1 = a|X = 0, p(I_1) = \epsilon/k) - \Pr(A_1 = a|X = 1, p(I_1) = \epsilon/k).\end{aligned}$$

If $p(I_1) = \epsilon/k$, the probability that exactly h elements are selected in this bin is at most $k/(m+k)(2s\epsilon/k)^h/h!$, and if they are selected, they are uniformly distributed in I_1 (although which of the sets p and q they are taken from is non-uniform). However, the probability that h elements are taken from I_1 is at least $\Omega(m/(m+k)(sm)^{-h}/h!)$ from the case where $p(I_1) = 1/m$, and in this case the elements are uniformly distributed in I_1 and uniformly from each of p and q. Therefore, we have that this contribution to our shared information is at most $k^2/(m(m+k))O(s\epsilon^2 m/k^2)^h/h!$. We note that $\epsilon^2 m/k^2 < 1$. Therefore, the sum of this over all $h \geq 3$ is $k^2/(m(m+k))O(s\epsilon^2 m/k^2)^3$. Summing over all $m + k$ bins, this is $k^{-4}\epsilon^6 s^3 m^2 = o(1)$.

It remains to analyze the case where $h = 2$. Once again, we have that ignoring which of p and q elements of A_1 came from, A_1 is identically distributed conditioned on $p(I_1) = 1/m$ and $|A_1| = 2$ as it is conditioned on $p(I_1) = \epsilon/k$ and $|A_1| = 2$. Since once again, the distributions $\mathcal{D}$ and $\mathcal{D}'$ are indistinguishable in the former case, we have that the contribution of the $h = 2$ terms to the shared information is at most

$$\begin{aligned}O\left(\frac{(k/(k+m)(\epsilon s/k)^2)^2}{m/(k+m)(s/m)^2}\right) d_{\mathrm{TV}}((A_1|X = 0, p(I_1)\epsilon/k, |A_1| = 2),\\ (A_1|X = 1, p(I_1) = \epsilon/k, |A_1| = 2))\end{aligned}$$

or

$$\begin{aligned}O\left(s^2 m k^{-2}\epsilon^4/(k+m)\right) d_{\mathrm{TV}}((A_1|X = 0, p(I_1) = \epsilon/k, |A_1| = 2),\\ (A_1|X = 1, p(I_1) = \epsilon/k, |A_1| = 2)) .\end{aligned}$$

It will suffice to show that conditioned upon $p(I_1) = \epsilon/k$ and $|A_1| = 2$ that

$$d_{\mathrm{TV}}((A_1|X = 0), (A_1|X = 1)) = O(1/\log(W)).$$

Let f be the order preserving linear function from $[0, 2]$ to I_1. Notice that conditional on $|A_1| = 2$ and $p(I_1) = \epsilon/k$ that we may sample from A_1 as follows:

- Pick two points $x > y$ uniformly at random from $[0, 2]$.
- Assign the points to p and q as follows:

- If $X = 0$ uniformly randomly assign these points to either distribution p or q.
- If $X = 1$ randomly do either:
  - Assign points in $[0, 1]$ to q and other points to p.
  - Assign points in $[0, 1]$ to p and other points to q.
- Randomly pick I_1 and apply f to x and y to get outputs $z = f(x), w = f(y)$.

Notice that the four cases:

(i) both points coming from p,
(ii) both points coming from q,
(iii) a point from p preceding a point from q,
(iv) a point from q preceding a point from p,

are all equally likely conditioned on either $X = 0$ or $X = 1$. However, we will note that this ordering is no longer independent of the choice of x and y.

Therefore, we can sample from A_1 subject to $X = 0$ and from A_1 subject to $X = 1$ in such a way that this ordering is the same deterministically. We consider running the above sampling algorithm to select (x, y) while sampling from $X = 0$ and (x', y') when sampling from $X = 1$ so that we are in the same one of the above four cases. We note that

$$d_{\mathrm{TV}}((A_1|X = 0), (A_1|X = 1)) \leq \mathbb{E}_{x,y,x',y'}[d_{\mathrm{TV}}((f(x), f(y)), (f(x'), f(y')))] \, ,$$

where the variation distance is over the random choices of f.

To show that this is small, we note that $|f(x) - f(y)|$ is distributed like $\ell_1(x - y)$. This means that $\log(|f(x) - f(y)|)$ is uniform over $[\log(f(x) - f(y)), \log(f(x) - f(y)) + 2\log(W)/3]$. Similarly, $\log(|f'(x') - f'(y')|)$ is uniform over $[\log(f(x') - f(y')), \log(f(x') - f(y')) + 2\log(W)/3]$. These differ in total variation distance by

$$O\left(\frac{|\log(f(x) - f(y))| + |\log(f(x') - f(y'))|}{\log(W)}\right) \, .$$

Taking the expectation over x, y, x', y' we get $O(1/\log(W))$. Therefore, we may further correlate the choices made in selecting our two samples, so that $z - w = z' - w'$ except with probability $O(1/\log(W))$. We note that after conditioning on this, z and z' are both uniformly distributed over subintervals of $[0, 2W]$ of length at least $2(W - W^{2/3})$. Therefore, the distributions on z and z' differ by at most $O(W^{-1/3})$. Hence, the total variation distance between A_1 conditioned on $|A_1| = 2, p(I_1) = \epsilon/k, X = 0$ and conditioned on $|A_1| = 2, p(I_1) = \epsilon/k, X = 1$ is at most $O(1/\log(W)) + O(W^{-1/3}) = O(1/\log(W))$. This completes our proof. ◀

We can now turn this into a lower bound for testing $\mathcal{A}_k$ distance on discrete domains.

Proof of second half of Theorem 3: Assume for sake of contradiction that this is not the case, and that there exists a tester taking $o(m)$ samples. We use this tester to come up with a continuous tester that violates Proposition 13.

We begin by proving a few technical bounds on the parameters involved. Firstly, note that we already have a lower bound of $\Omega(k^{1/2}/\epsilon^2)$, so we may assume that this is much less than m. We now claim that $m = O(\min(k^{2/3}\log^{1/3}(3 + n/(m + k))/\epsilon^{4/3}, k^{4/5}/\epsilon^{6/5})$. If $m \leq k$, there is nothing to prove. Otherwise,

$$k^{2/3}\log^{1/3}(3 + n/(m + k))/\epsilon^{4/3} \geq m(m/k)^{-1/3}\log(3 + n/(m + k))^{1/3}.$$

Thus, there is nothing more to prove unless $\log(3 + n/(m + k)) \gg m/k$. But, in this case, $\log(3 + n/(m + k)) \gg \log(m/k)$ and thus $\log(3 + n/(m + k)) = \Theta(\log(3 + n/k))$, and we are done.

We now let $W = n/(6(m+k))$, and let $\mathcal{D}$ and $\mathcal{D}'$ be as specified in Proposition 13. We claim that we have a tester to distinguish a p, q from $\mathcal{D}$ from ones taken from $\mathcal{D}'$ in $o(m)$ samples. We do this as follows: By rounding p and q down to the nearest third of an integer, we obtain p',q' supported on set of size n. Since p and q were piecewise constant on pieces of size at least 1, it is not hard to see that $\|p' - q'\|_{\mathcal{A}_k} \geq \|p - q\|_{\mathcal{A}_k}/3$. Therefore, a tester to distinguish $p' = q'$ from $\|p' - q'\|_{\mathcal{A}_k} \geq \epsilon$ can be used to distinguish $p = q$ from $\|p - q\|_{\mathcal{A}_k} \geq 3\epsilon$. This is a contradiction and proves our lower bound. ◀

References

1. J. Acharya, I. Diakonikolas, C. Hegde, J. Li, and L. Schmidt. Fast and Near-Optimal Algorithms for Approximating Distributions by Histograms. In *34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2015*, pages 249–263, 2015.
2. J. Acharya, I. Diakonikolas, J. Li, and L. Schmidt. Fast algorithms for segmented regression. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, pages 2878–2886, 2016.
3. J. Acharya, I. Diakonikolas, J. Li, and L. Schmidt. Sample-optimal density estimation in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1278–1289, 2017. Full version available at https://arxiv.org/abs/1506.00671.
4. R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk. *Statistical Inference under Order Restrictions*. Wiley, New York, 1972.
5. T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *IEEE Symposium on Foundations of Computer Science*, pages 259–269, 2000. URL: `citeseer.ist.psu.edu/batu00testing.html`.
6. T. Batu, R. Kumar, and R. Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *ACM Symposium on Theory of Computing*, pages 381–390, 2004.
7. C. L. Canonne. A survey on distribution testing: Your data is big. but is it blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22:63, 2015.
8. C. L. Canonne. Are few bins enough: Testing histogram distributions. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016*, pages 455–463, 2016.
9. C. L. Canonne, I. Diakonikolas, T. Gouleakis, and R. Rubinfeld. Testing shape restrictions of discrete distributions. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*, pages 25:1–25:14, 2016.
10. S. Chan, I. Diakonikolas, R. Servedio, and X. Sun. Learning mixtures of structured distributions over discrete domains. In *SODA*, pages 1380–1394, 2013.
11. S. Chan, I. Diakonikolas, R. Servedio, and X. Sun. Efficient density estimation via piecewise polynomial approximation. In *STOC*, pages 604–613, 2014.
12. S. Chan, I. Diakonikolas, R. Servedio, and X. Sun. Near-optimal density estimation in near-linear time using variable-width histograms. In *NIPS*, pages 1844–1852, 2014.
13. S. Chan, I. Diakonikolas, P. Valiant, and G. Valiant. Optimal algorithms for testing closeness of discrete distributions. In *SODA*, pages 1193–1203, 2014.
14. S. Chaudhuri, R. Motwani, and V. R. Narasayya. Random sampling for histogram construction: How much is enough? In *SIGMOD Conference*, pages 436–447, 1998.
15. C. Daskalakis, A. De, G. Kamath, and C. Tzamos. A size-free CLT for poisson multinomials and its applications. In *Proceedings of the 48th Annual ACM Symposium on the Theory of Computing*, STOC'16, New York, NY, USA, 2016. ACM.

16 C. Daskalakis, I. Diakonikolas, R. O'Donnell, R. A. Servedio, and L. Tan. Learning Sums of Independent Integer Random Variables. In *FOCS*, pages 217–226, 2013.

17 C. Daskalakis, I. Diakonikolas, R. Servedio, G. Valiant, and P. Valiant. Testing k-modal distributions: Optimal algorithms via reductions. In *SODA*, pages 1833–1852, 2013.

18 C. Daskalakis, I. Diakonikolas, and R. A. Servedio. Learning k-modal distributions via testing. In *SODA*, pages 1371–1385, 2012.

19 C. Daskalakis, I. Diakonikolas, and R. A. Servedio. Learning Poisson Binomial Distributions. In *STOC*, pages 709–728, 2012.

20 L. Devroye and G. Lugosi. *Combinatorial methods in density estimation*. Springer Series in Statistics, Springer, 2001.

21 L. Devroye and G. Lugosi. Bin width selection in multivariate histograms by the combinatorial method. *Test*, 13(1):129–145, 2004.

22 I. Diakonikolas, T. Gouleakis, J. Peebles, and E. Price. Collision-based testers are optimal for uniformity and closeness. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:178, 2016.

23 I. Diakonikolas, M. Hardt, and L. Schmidt. Differentially private learning of structured discrete distributions. In *NIPS*, pages 2566–2574, 2015.

24 I. Diakonikolas and D. M. Kane. A new approach for testing properties of discrete distributions. In *FOCS*, pages 685–694, 2016. Full version available at abs/1601.05557.

25 I. Diakonikolas, D. M. Kane, and V. Nikishkin. Optimal algorithms and lower bounds for testing closeness of structured distributions. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 1183–1202, 2015.

26 I. Diakonikolas, D. M. Kane, and V. Nikishkin. Testing identity of structured distributions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1841–1854, 2015.

27 I. Diakonikolas, D. M. Kane, and A. Stewart. Efficient robust proper learning of log-concave distributions. *CoRR*, abs/1606.03077, 2016.

28 I. Diakonikolas, D. M. Kane, and A. Stewart. The fourier transform of poisson multinomial distributions and its algorithmic applications. In *Proceedings of STOC'16*, 2016.

29 I. Diakonikolas, D. M. Kane, and A. Stewart. Learning multivariate log-concave distributions. *CoRR*, abs/1605.08188, 2016.

30 I. Diakonikolas, D. M. Kane, and A. Stewart. Optimal Learning via the Fourier Transform for Sums of Independent Integer Random Variables. In *COLT*, volume 49, pages 831–849, 2016. Full version available at arXiv:1505.00662.

31 I. Diakonikolas, D. M. Kane, and A. Stewart. Properly learning poisson binomial distributions in almost polynomial time. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016*, pages 850–878, 2016. Full version available at arXiv:1511.04066.

32 D. Freedman and P. Diaconis. On the histogram as a density estimator:l2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(4):453–476, 1981.

33 A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *STOC*, pages 389–398, 2002.

34 P. Groeneboom and G. Jongbloed. *Nonparametric Estimation under Shape Constraints: Estimators, Algorithms and Asymptotics*. Cambridge University Press, 2014.

35 S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Trans. Database Syst.*, 31(1):396–438, 2006.

36 P. Indyk, R. Levi, and R. Rubinfeld. Approximating and Testing k-Histogram Distributions in Sub-linear Time. In *PODS*, pages 15–22, 2012.

37 H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, pages 275–286, 1998.

38 J. Klemela. Multivariate histograms with data-dependent partitions. *Statistica Sinica*, 19(1):159–176, 2009.

39 E. L. Lehmann and J. P. Romano. *Testing statistical hypotheses*. Springer Texts in Statistics. Springer, 2005.

40 G. Lugosi and A. Nobel. Consistency of data-driven histogram methods for density estimation and classification. *Ann. Statist.*, 24(2):687–706, 04 1996.

41 J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933. doi:10.1098/rsta.1933.0009.

42 L. Paninski. A coincidence-based test for uniformity given very sparsely-sampled discrete data. *IEEE Transactions on Information Theory*, 54:4750–4755, 2008.

43 R. Rubinfeld. Taming big probability distributions. *XRDS*, 19(1):24–28, 2012.

44 D. W. Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, 1979.

45 D. W. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. Wiley, New York, 1992.

46 N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *SIGMOD Conference*, pages 428–439, 2002.

47 G. Valiant and P. Valiant. An automatic inequality prover and instance optimal identity testing. In *FOCS*, 2014.

48 R. Willett and R. D. Nowak. Multiscale poisson intensity and density estimation. *IEEE Transactions on Information Theory*, 53(9):3171–3187, 2007.

Deleting and Testing Forbidden Patterns in Multi-Dimensional Arrays

Omri Ben-Eliezer[1], Simon Korman[2], and Daniel Reichman[3]

1 **Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
omribene@gmail.com
2 **Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel**
simon.korman@gmail.com
3 **Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA**
daniel.reichman@gmail.com

Abstract

Analyzing multi-dimensional data is a fundamental problem in various areas of computer science. As the amount of data is often huge, it is desirable to obtain sublinear time algorithms to understand local properties of the data.

We focus on the natural problem of testing *pattern freeness*: given a large d-dimensional array A and a fixed d-dimensional pattern P over a finite alphabet Γ, we say that A is *P-free* if it does not contain a copy of the forbidden pattern P as a consecutive subarray. The distance of A to P-freeness is the fraction of the entries of A that need to be modified to make it P-free.

For any $\epsilon > 0$ and any large enough pattern P over any alphabet – other than a very small set of exceptional patterns – we design a *tolerant tester* that distinguishes between the case that the distance is at least ϵ and the case that the distance is at most $a_d\epsilon$, with query complexity and running time $c_d\epsilon^{-1}$, where $a_d < 1$ and c_d depend only on the dimension d. These testers only need to access uniformly random blocks of samples from the input A.

To analyze the testers we establish several combinatorial results, including the following d-dimensional *modification lemma*, which might be of independent interest: For any large enough d-dimensional pattern P over any alphabet (excluding a small set of exceptional patterns for the binary case), and any d-dimensional array A containing a copy of P, one can delete this copy by modifying one of its locations without creating new P-copies in A.

Our results address an open question of Fischer and Newman, who asked whether there exist efficient testers for properties related to tight substructures in multi-dimensional structured data.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Property testing, Sublinear algorithms, Pattern matching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.9

1 Introduction

Pattern matching is the algorithmic problem of finding occurrences of a fixed pattern in a given string. This problem appears in many settings and has applications in diverse domains such as computational biology, computer vision, natural language processing and web search. There has been extensive research concerned with developing algorithms that search for patterns in strings, resulting with a wide range of efficient algorithms [12, 24, 19, 14, 26, 25]. Higher-dimensional analogues where one searches for a d-dimensional pattern in a d-dimensional

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 9; pp. 9:1–9:14

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

array have received attention as well. For example, the 2D case arises in analyzing aerial photographs [7, 8] and the 3D case has applications in medical imaging.

Given a string S of length n and a pattern P of length $k \leq n$, any algorithm which determines whether P occurs in S has running time $\Omega(n)$ [13, 29] and a linear lower bound carries over to higher dimensions. For the 2D case, when an $n \times n$ image is concerned, algorithms whose run time is $O(n^2)$ are known [8]. These algorithms have been generalized to the 3D case to yield running time of $O(n^3)$ [18] for $n \times n \times n$ arrays. Finally it is also known (e.g., [21]) that for the d-dimensional case it is possible to solve the pattern matching problem in time $O(d^2 n^d \log m)$ (where the pattern is an array of size m^d). It is natural to ask which tasks of this type can be performed in sublinear (namely $o(n^d)$) time for d-dimensional arrays.

The field of *property testing* [20, 30] deals with decision problems regarding discrete objects (e.g., graphs, functions, images) that either satisfy a certain property $\mathcal{P}$ or are *far* from satisfying $\mathcal{P}$. Here, the property $\mathcal{P}$ consists of all d-dimensional arrays that avoid a predetermined pattern P. *Tolerant* property testing [27] is a useful extension of the standard notion, in which the tester needs to distinguish between objects that are *close* to satisfying the property and those that are *far* from satisfying it.

A d-dimensional $k_1 \times \ldots \times k_d$ array A over an alphabet Γ is a function from $[k_1] \times \ldots \times [k_d]$ to Γ, where for an integer $k > 0$ we let $[k]$ denote the set $\{0, \ldots, k-1\}$ and write $[k]^d = [k] \times \ldots \times [k]$. For simplicity of presentation, all results in this paper are presented for *square* arrays in which $k_1 = \ldots = k_d$, but they generalize to non-square arrays in a straightforward manner. We consider the (tolerant) *pattern-freeness problem*. An (ϵ_1, ϵ_2)-tester Q for this problem is a randomized algorithm that is given access to a d-dimensional array A, as well as its size and proximity parameters $0 \leq \epsilon_1 < \epsilon_2 < 1$. Q needs to distinguish with probability at least $2/3$ between the case that A is ϵ_1-close to being P-free and the case that A is ϵ_2-far from being P-free. The query complexity of Q is the number of queries it makes in A.

Our interest in the pattern-freeness problem stems from several applications. In certain scenarios of interest, we might be interested in identifying quickly that an array is far from not containing a given pattern. In the one dimensional case, being far from not containing a given text may indicate a potential anomaly which requires attention (e.g., an offensive word in social network media), hence such testing algorithms may provide useful in anomaly detection. Many computer vision methods for classifying images are feature based; being far from not containing a certain pattern associated with a feature may be useful in rejection methods that enable us to quickly discard images that do not possess a certain visual property.

Beyond practical applications, devising property testing algorithms for the pattern freeness problem is of theoretical interest. In the first place, it leads to a combinatorial characterization of the distance from being P-free. Such a characterization has proved fruitful in graph property testing [3, 4] where celebrated graph removal lemmas were developed en route of devising algorithms for testing subgraph freeness. We encounter a similar phenomena in studying patterns and arrays: at the core of our approach for testing pattern freeness lies a *modification lemma* for patterns which we state next. We believe that this lemma may be of independent interest and find applications beyond testing algorithms.

For a pattern P of size $k \times k \times \ldots \times k$, an entry whose location in $\mathcal{P}$ is in $\{0, k-1\} \times \ldots \times \{0, k-1\}$ is said to be a *corner* of P. We say that P is *almost homogeneous* if all of its entries but one are equal, and the different entry lies in a corner of P. Finally, P is *removable* (with respect to the alphabet Γ) if for any d-dimensional array A over Γ and any copy of P in A, one can destroy the copy by modifying one of its entries without creating new P-copies in A. The modification lemma states that for any d, and any large enough pattern P, when the

alphabet is binary it holds that P is removable *if and only if* it is not almost homogeneous, and when the alphabet is not binary, P is removable provided that it is large enough.

Recent works [10, 11] have obtained tolerant testers for visual properties. As observed in these works, tolerance is an attractive property for testing visual properties as real-world images are often noisy. With the modification lemma at hand, we show that when P is removable, the (relative) *hitting number* of P in A, which is the minimal size of a set of entries that intersects all P-copies in A divided by $|A|$, differs from the distance of A from P-freeness by a multiplicative factor that depends only on the dimension d of the array. This relation allows us to devise very fast $(5^{-d}\epsilon, \epsilon)$-tolerant testers for P-freeness, as the hitting number of P in A can be well approximated using only a very small sample of blocks of entries from A. The query complexity of our tester is c_d/ϵ, where c_d is a positive constant depending only on d. Note that our characterization in terms of the hitting number is crucial: Merely building on the fact that A contains many occurrences of P (as can be derived directly from the modification lemma) and randomly sampling $O(1/\epsilon)$ possible locations in A, checking whether the sub-array starting at these locations equals P, would lead to query complexity of $O(k^d/\epsilon)$. Note that our tester is optimal (up to a multiplicative factor that depends on d), as any tester for this problem must make $\Omega(1/\epsilon)$ queries.

The one dimensional setting, where one seeks to determine quickly whether a string S is ϵ-far from being P-free is of particular interest. We are able to leverage the modification lemma and show that the distance of a string S from being P-free for a fixed pattern P (that is not almost homogeneous) is *exactly equal* to the hitting number of P in A. For an arbitrary constant $0 < c < 1$, this characterization allows us to devise a $((1-c)\epsilon, \epsilon)$-tolerant tester making $O_c(1/\epsilon)$ queries for this case. For the case of almost homogeneous patterns, and an arbitrary constant $c > 0$, we devise a $((1/(16+c))\epsilon, \epsilon)$-tolerant tester that makes $O_c(1/\epsilon)$ queries. Whether tolerant testers exist for almost homogenous patterns of dimension larger than 1 is an open question.

2 Related Work

The problem of testing pattern freeness is related to the study of testing subgraph freeness (see, for example, [1, 4, 2]). This line of work examines how one can test quickly whether a given graph G is H-free or ϵ-far from being H-free, where H is a fixed subgraph. In this problem, a graph is ϵ-far from being H-free if at least an ϵ-fraction of its edges and non-edges need to be altered in order to ensure that the resulting graph does not contain H as a (not necessarily induced) subgraph. A key component in these works are removal lemmas: typically such lemmas imply that if G is ϵ-far from being H-free, then it contains a large number of copies of H. For example, the *triangle removal lemma* asserts that for every $\epsilon > 0$, there exists $\delta = \delta(\epsilon) > 0$ such that if an n-vertex graph G is ϵ-far from being triangle free, then G contains at least δn^3 triangles (see, e.g., [6] and the references within).

Alon *et. al.* [3] studied the problem of testing regular languages. Testing pattern-freeness (1-dimensional, binary alphabet, constant pattern length k) is a special case of the former, since the language L of all strings avoiding a fixed pattern is regular. The query complexity of their tester is $O\left(\frac{s^3}{\epsilon} \cdot \ln^3(\frac{1}{\epsilon})\right)$, where s is the minimal size of a DFA that accepts the regular language L. In the case of the regular language considered here a simple pumping-lemma inspired argument shows that $s \geq \Omega(k)$. Hence the upper bound on testing pattern freeness implied by their algorithm is $O\left(\frac{k^3}{\epsilon} \cdot \ln^3(\frac{1}{\epsilon})\right)$. Our 1D tester solves a very restricted case of the problem the tester of [3] deals with, but it achieves a query complexity of $O(1/\epsilon)$ in this setting. Moreover, our tester is much simpler and can be applied in the more general high

dimensional setting, or when the pattern length k is allowed to grow as a function of the string length n.

The problem of testing *submatrix freeness* is investigated in [15, 16, 5, 17, 2]. As opposed to our case, which is concerned with *tight* submatrices, all of these results deal with submatrices that are not necessarily tight (i.e. the rows and the columns need not be consecutive). Quantitatively, the submatrix case is very different from our case: in our case P-freeness can be testable using $O(\epsilon^{-1})$ queries, while in the submatrix case, for a binary submatrix of size $k \times k$ a lower bound of $\epsilon^{-\Omega(k^2)}$ on the needed number of queries is easy to obtain, and in the non-binary case there exist 2×2 matrices for which there exists a super polynomial lower bound of $\epsilon^{-\Omega(\log 1/\epsilon)}$.

The 2D part of our work adds to a growing literature concerned with testing properties of images [28, 31, 10]. Ideas and techniques from the property testing literature have recently been used in the fields of computer vision and pattern recognition [22, 23].

3 Notation and definitions

An (n, d)-*array* A over an alphabet Γ is a function from $[n]^d$ to Γ. The $x = (x_1, \ldots, x_d)$ *entry* of A, denoted by A_x, is the value of the function A at location x. Let P be a (k, d)-array over an alphabet Γ of size at least two. We say that a d-dimensional array A *contains* a copy of P (or a P-copy) *starting in location* $x = (x_1, \ldots, x_d)$ if for any $y \in [k]^d$ we have $A_{x+y} = P_y$. Finally, A is P-free if it does not contain copies of P.

A property $\mathcal{P}$ of d-dimensional arrays is simply a family of such arrays over an alphabet Γ. For an array A and a property $\mathcal{P}$, the *absolute distance* $d_{\mathcal{P}}(A)$ of A to $\mathcal{P}$ is the minimal number of entries that one needs to change in A to get an array that satisfies $\mathcal{P}$. The *relative distance* of A to $\mathcal{P}$ is $\delta_{\mathcal{P}}(A) = d_{\mathcal{P}}(A)/|A|$, where clearly $0 \leq \delta_{\mathcal{P}}(A) \leq 1$ for any nontrivial $\mathcal{P}$ and A. We say that A is ϵ-close to $\mathcal{P}$ if $\delta_{\mathcal{P}}(A) \leq \epsilon$, and ϵ-far if $\delta_{\mathcal{P}}(A) \geq \epsilon$. In this paper we consider the property of P-freeness, which consists of all P-free arrays. The absolute and relative distance to P-freeness are denoted by $d_P(A)$ and $\delta_P(A)$, respectively.

For an array A and a pattern P, a *deletion set* is a set of entries in A whose modification can turn it to be P-free, and $d_P(A)$ is called the *deletion number*, since it is the size of a minimal deletion set. Similarly, a given set of entries in A is a *hitting set* if every P-copy in A contains at least one of these entries. The *hitting number* $h_P(A)$ is the size of the minimal hitting set for P in A. For all notation here and above, in the 1-dimensional case we replace A by S (for String).

We use several definitions from [27]. Let $\mathcal{P}$ be a property of arrays and let $h_1, h_2 : [0, 1] \to [0, 1]$ be two monotone increasing functions. An (h_1, h_2)-*distance approximation* algorithm for $\mathcal{P}$ is given query access to an unknown array A. The algorithm outputs an estimate $\hat{\delta}$ to $\delta_P(A)$, such that with probability at least $2/3$ it holds that $h_1(\delta_P(A)) \leq \hat{\delta} \leq h_2(\delta_P(A))$. For a property $\mathcal{P}$ and for $0 \leq \epsilon_1 < \epsilon_2 \leq 1$, an (ϵ_1, ϵ_2)-*tolerant tester* for $\mathcal{P}$ is given query access to an array A. The tester *accepts* with probability at least $2/3$ if A is ϵ_1-close to $\mathcal{P}$, and *rejects* with probability at least $2/3$ if A is ϵ_2-far from $\mathcal{P}$. In the "standard" notion of property testing, $\epsilon_1 = 0$. Thus, any tolerant tester is also a tester in the standard notion. Finally, we define the additive (multiplicative) *tolerance* of the tester above as $\epsilon_2 - \epsilon_1$ (ϵ_2/ϵ_1 respectively).

Table 1 Summary of results. $0 < \tau < 1$ and $c > 0$ are arbitrary constants. α_c is a constant that depends only on c. $\beta_{d,\tau}$ is a constant that depends only on d and τ. 'modification lemma' specifies if patterns are classified as removable or not. The 'tester tolerance' is multiplicative.

dim.	template type	modification lemma	tester tolerance	query complexity
1D	general	removable for any k	$1/(1-\tau)$	$O(1/\epsilon\tau^3)$
	almost homog.	not removable for any k	$16+c$	α_c/ϵ
2^+D	general	removable for $k > 3 \cdot 2^d$	$(1-\tau)^{-d}\alpha_d$	$\beta_{d,\tau}/\epsilon$
	almost homog.	not removable for any k	–	–

4 Main Results

The modification lemma presented is central in the study of minimal deletion sets. It classifies the possible patterns into ones that are removable and ones that are not. The result that the vast majority of patterns are removable is used extensively throughout the paper in the design and proofs of algorithms for efficient testing of pattern freeness (in 1 and higher dimensions).

Our 1-dimensional modification lemma (Lemma 1) gives the following full characterization of 1-dimensional patterns (i.e. strings). A binary pattern is removable *if and only if* it is not almost homogeneous, while *any* pattern over a larger alphabet is removable. The multidimensional version of the lemma (Lemma 2) makes the exact same classification, but for (k,d)-arrays for which $k \geq 3 \cdot 2^d$.

The fact that most patterns are removable is very important for analyzing the deletion number. For example, observe that a removable pattern appears at least $d_P(A)$ times (possibly with overlaps) in the array A, so an ϵ-tester can simply check for the presence of the d-dimensional pattern in $1/\epsilon$ random locations in the array, using $O(k^d/\epsilon)$ queries.

Another important part of our work makes explicit connections between the deletion number and the hitting number for both 1 and higher dimensions. These are needed in order to get improved testers (e.g. for getting rid of k in the sample complexity) in d-dimensions.

In the 1-dimensional removable case we show that the deletion number $d_P(S)$ equals the hitting number $h_P(S)$. We derive a $((1-\tau)\epsilon,\epsilon)$-tolerant tester for any fixed $\tau > 0$ and any $0 < \epsilon \leq 1$, whose number of queries and running time are $O(\epsilon^{-1}\tau^{-3})$ (Corollary 13).

For higher dimensions, we show (Lemma 11) that $h_P(A) \leq d_P(A) \leq \alpha_d h_P(A) \leq \alpha_d k^{-d}$, where $\alpha_d = 4^d + 2^d$ depends only on the dimension d. This bound gives a $((1-\tau)^d\alpha_d^{-1}\epsilon,\epsilon)$-tolerant tester making $C_\tau\epsilon^{-1}$ queries, where $C_\tau = O(1/\tau^d(1-(1-\tau)^d)^2)$ (Theorem 15). The running time here is $C'_\tau\epsilon^{-1}$ where C'_τ depends only on τ.

In the full version of the paper [9], for the 1-dimensional setting we also provide dedicated algorithms to handle the almost homogeneous (non-removable) patterns, obtaining an $O(n)$ algorithm for computing the deletion number as well as an $(\epsilon/(16+c),\epsilon)$-tolerant tester, for any constant $c > 0$, using $\alpha_c\epsilon^{-1}$ queries, where α_c depends only on c.

Finally, we provide a lower bound of $\Omega(1/\epsilon)$ (full proof in can be found in [9]) for any general tester of pattern freeness. Our main results are summarized in Table 1.

A natural question is what happens if one is concerned with pattern freeness with respect to several patterns simultaneously. Namely, testing quickly whether an array satisfies the property of not containing a fixed set of patterns $P_1,\ldots,P_r$ with $r > 1$, or is far from satisfying this property. Our results do not apply to this setting, with the main obstacle being our modification lemmas. Namely, the difficulty is that for several distinct patterns $P_1 \ldots P_r$, modifying an occurrence of P_i may create an new occurrence of P_j (where $i \neq j$).

5 Modification Lemma

We begin with the proof of the 1-dimensional modification lemma. The main strategy here is to consider the longest streaks of zeros and ones (0s and 1s) in the pattern - a strategy that cannot be used in higher dimensions.

▶ **Theorem 1** (1D Modification Lemma). *For a 1-dimensional pattern P over an alphabet Γ:*

1. *If $|\Gamma| = 2$ then P is removable if and only if it is not almost homogeneous.*
2. *If $|\Gamma| \geq 3$ then P is removable.*

▶ Remark. The second statement of the theorem can be easily derived from the first statement. If P does not contain all letters in Γ then it is clearly removable, as changing any of its entries to any of the missing letters cannot create new P-copies. Otherwise, we can reduce the problem to the binary case: let σ_1, σ_2 be the letters in Γ that appear the smallest number of times in P (specifically, if P is of length 3 and has exactly three different letters, we pick σ_1 and σ_2 to be the first letter and the last letter in P, respectively). Consider the following pattern P' over $\{0,1\}$: $P'_x = 0$ if $P_x \in \{\sigma_1, \sigma_2\}$ and $P'_x = 1$ otherwise. Observe that P' is not almost homogeneous, implying that it is removable. It is not hard to verify now that P is removable as well.

Proof of Theorem 1. As discussed above, it is enough to consider the binary case. Let $P = P_0 \dots P_{k-1}$ be a binary pattern of length k that is not almost homogeneous, and let S be an arbitrary binary string containing P. We need to show that one can flip one of the bits of P without creating a new P-copy in S. We assume that P contains both 0s and 1s (i.e. it is not homogeneous) otherwise flipping any bit would work. We may assume that $k \geq 3$ (since for $k = 1, 2$ all patterns are homogeneous or almost homogeneous). Let us also assume that P starts with a 1, i.e. $P_0 = 1$ and let $t \leq k-1$ be the length of the longest 0-streak (sub-string of consecutive 0s) in P. Let $i > 0$ be the leftmost index in which such a 0-streak of length t begins. Clearly, $P_{i-1} = 1$ and $P_i = \dots = P_{i+t-1} = 0$.

If $i + t \leq k - 1$ (i.e. the streak is not at the end of P) then $P_{i+t} = 1$ and in such a case if we modify P_{i+t} to 0, the copy of P is removed without creating new P-copies in S. To see this, observe that a new copy cannot start at the bit flip location $i + t$ or within the 0-streak at any of its locations $i, \dots, i + t - 1$ since the bits in these locations are 0 while the starting bit of P is 1. Note that flipping P_{i+t} does not create new P-copies starting after the location of P_{i+t} in S. Furthermore, no new copy starting before P_i is created since otherwise it would contain a 0-streak of length $t + 1$.

Thus, we may assume that P contains exactly one 0-streak of length t, lying at its last t locations, so $P_{k-1} = 0$. Denote by s the length of the longest 1-streak in P; a symmetric reasoning shows that P begins with its only 1-streak of length s. If P is *not* of the form 1^s0^t, it can be verified that flipping P_s (the leftmost 0 in P) to 1 does not create any P-copy. The only case left is $P = 1^s0^t$, where $s, t \geq 2$ since P is not almost homogeneous. Consider the bit of the string S that is to the left of P. If it is a 0 then we flip P_1 to 0 and otherwise, we flip P_0 to 0, where in both cases no new copy is created. ◀

We now turn to proving the high dimensional version of the modification lemma. Here, as opposed to the 1-dimensional case, there exist patterns that are neither removable nor almost-homogeneous. However, we show that if a pattern is large enough and not almost homogeneous then it is removable.

▶ **Theorem 2** (Modification Lemma). *Let $d > 1$ and let P be a (k, d)-array over the alphabet Γ where $k \geq 3 \cdot 2^d$.*

1. *If $|\Gamma| = 2$ then P is removable if and only if it is not almost homogeneous.*
2. *If $|\Gamma| \geq 3$ then P is removable.*

▶ Remark. Theorem 2 states that any large enough binary pattern which is not almost homogeneous is removable. The requirement that the pattern is large enough is crucial, as can be seen from the following 2-dimensional example. The 2×2 pattern $P = [01; 01]$ is in the center of the 4×4 matrix $A = [0101; 0011; 0011; 0101]$. Flipping any bit in P creates a new P-copy in A. This example easily generalizes to a $2 \times \ldots \times 2$ pattern in a $4 \times \ldots \times 4$ array, while the dependence of k on d is exponential in the statement of Theorem 2. It will be interesting to understand what is the correct order of magnitude of this dependence.

Proof of Theorem 2. The reduction from a general alphabet to a binary one, described after the statement of Theorem 1, can be used here as well. Thus, it is enough to prove the first statement of the theorem. If P is binary and almost homogeneous then it is not removable: Without loss of generality $P_{(0,\ldots,0)} = 1$ and $P_x = 0$ for any $x \neq (0, \ldots, 0)$. Consider a $(2k, d)$-array A such that $M_{(0,\ldots,0)} = M_{(1,\ldots,1)} = 1$ and $A = 0$ elsewhere. Modifying any bit of the P-copy starting at $(1, \ldots, 1)$ creates a new P-copy in A, hence P is not removable.

The rest of the proof is dedicated to the other direction. Suppose that P is a binary (k, d)-array that is not removable. We would like to show that P must be almost homogeneous. As P is not removable, there exists a binary array A containing a copy of P such that flipping any single bit in this copy creates a new copy of P in A. This copy of P will be called the *template* of P in A.

Clearly, all of the new copies created by flipping bits in the template must intersect the template, so we may assume that A is of size $(3k - 2)^d$ and that the template starts in location $\tilde{k} = (k - 1, \ldots, k - 1)$. For convenience, let $I = [k]^d$ denote the set of indices of P. For any $x \in I$ let $\bar{x} = x + \tilde{k}$; $\bar{x}$ is the location in A of bit x of the template.

Roughly speaking, our general strategy for the proof is to show that there exist at most two "special" entries in P such that when we flip a bit in the template (creating a new copy of P in A) the flipped bit usually plays the role of one of the special entries in the new copy. We then show that in fact, there must be exactly one special entry, which must lie in a corner of P, and that all non-special entries are equal while the special entry is equal to their negation. This will finish the proof that P is almost homogeneous.

▶ **Definition 3.** Let $i \leq d$ and let δ be positive integers. Let $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ be d-dimensional points. The pair (x, y) is (i, δ)-*related* if $y_i - x_i = \delta$ and $y_j = x_j$ for any $j \neq i$. An (i, δ)-related pair (x, y) is said to be an (i, δ)-*jump* in P if $P_x \neq P_y$.

In other words, (x, y) is (i, δ)-*related* if there is an increasing path of length δ along coordinate i in the hypergrid graph on $[n]^d$.

▶ **Lemma 4.** *For any $1 \leq i \leq d$ there exists $0 < \Delta_i < k/3$ such that at most two of the (i, Δ_i)-related pairs of points from I are (i, Δ)-jumps in P.*

Proof. Recall that, by our assumption, flipping any of the $K = k^d$ bits of the template creates a new copy of P in A. Consider the following mapping $m : I \to [2k-1]^d$. $m(x_1, \ldots, x_d)$ is the starting location of a new copy of P created in A as a result of flipping bit $x = (x_1, \ldots, x_d)$ of the template (which is bit $\bar{x}$ of A). If more than one copy is created by this flip, then we choose the starting location of one of the copies arbitrarily.

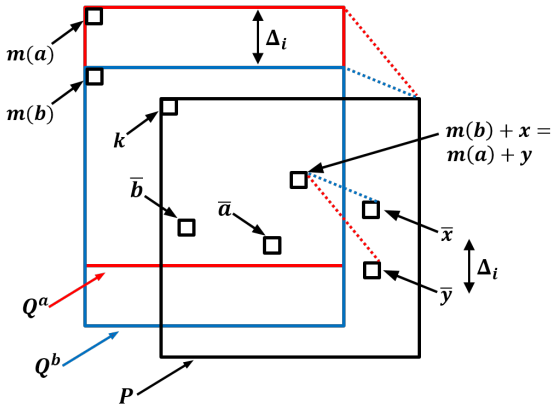


Figure 1 Illustration for Lemma 4. A 2-dimensional example, where i is the vertical coordinate: Flipping the bit (of the template P) at location $\bar{a}$ creates the P-copy Q^a at location $m(a)$. Similarly, the copy Q^b is created at location $m(b)$. Note that the pair of points $(\bar{x}, \bar{y})$ (which is (x, y) in P) and the copy locations pair $(m(a), m(b))$ are both (i, Δ_i)-related. The values P_x and P_y ($M_{\bar{x}}$ and $M_{\bar{y}}$) must be equal.

Observe that m is injective, and let S be the image of m, where $|S| = K$. Let $1 \leq i \leq d$ and consider the collection of (1-dimensional) lines

$$\mathcal{L}_i = \big\{\{x_1\} \times \ldots \times \{x_{i-1}\} \times [2k-1] \times \{x_{i+1}\} \times \ldots \times \{x_d\} \quad | \quad \forall j \neq i : x_j \in [2k-1]\big\}.$$

Clearly $\sum_{\ell \in \mathcal{L}_i} |S \cap \ell| = K$. On the other hand, $|\mathcal{L}_i| = \prod_{j \neq i}(2k-1) < 2^{d-1} \prod_{j \neq i} k = 2^{d-1}K/k$, so there exists a line $\ell \in \mathcal{L}_i$ for which $|S \cap \ell| > k/2^{d-1} \geq 6$. Hence $|S \cap \ell| \geq 7$. Let $\alpha_1 < \ldots < \alpha_7$ be the smallest i-indices of elements in $S \cap \ell$. Since $\alpha_7 - \alpha_1 < 2k-1$ there exists some $1 \leq l \leq 6$ such that $\alpha_{l+1} - \alpha_l < k/3$. That is, S contains an (i, Δ_i)-related pair with $0 < \Delta_i < k/3$. In other words, there are two points $a, b \in I$ such that flipping $\bar{a}$ ($\bar{b}$) would create a new P-copy, denoted by Q^a (Q^b respectively), which starts in location $m(a)$ ($m(b)$ respectively) in A, and $(m(a), m(b))$ is an (i, Δ_i)-related pair.

The following claim finishes the proof of the lemma and will also be useful later on.

▶ **Claim 5.** *For a and b as above, let (x, y) be a pair of points from I that are (i, Δ_i)-related and suppose that $y \neq \bar{a} - m(a)$ and that $x \neq \bar{b} - m(b)$. Then $P_x = P_y$.*

Proof. The bits that were flipped in A to create Q^a and Q^b are $\bar{a}, \bar{b}$ respectively. Since $y + m(a) \neq \bar{a}$, the entry $M_{y+m(a)}$ serves as the entry of the P-copy Q^a in location y, so $P_y = M_{y+m(a)}$. Similarly, since $x + m(b) \neq \bar{b}$, we have $P_x = M_{x+m(b)}$. But since both pairs (x, y) and $(m(a), m(b))$ are (i, Δ_i)-related, we get that $m(b) - m(a) = y - x$, implying that $x + m(b) = y + m(a)$, and therefore $P_x = M_{x+m(b)} = M_{y+m(a)} = P_y$, as desired. ◀

Clearly, the number of (i, Δ_i)-related pairs that do not satisfy the conditions of the claim is at most two, finishing the proof of Lemma 4. ◀

Let $\Delta = (\Delta_1, \ldots, \Delta_d)$ where for any $1 \leq i \leq d$, we take Δ_i that satisfies the statement of Lemma 4 (its specific value will be determined later).

▶ **Definition 6.** Let $x \in I$. The set of Δ-*neighbours* of x is

$$N_x = \big\{y \in I \quad | \quad \exists i : (x, y) \text{ is } (i, \Delta_i)\text{-related or } (y, x) \text{ is } (i, \Delta_i)\text{-related}\big\}$$

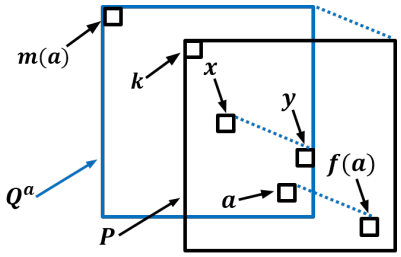


Figure 2 Illustration for Definition 8. Recall that flipping a bit $\bar{a}$ in A creates a new P-copy Q^a (which contains $\bar{a}$), located at the point $m(a)$ in the coordinates of A. The bits x and a are *mapped* to y and $f(a)$ respectively.

and the number of Δ-neighbours of x is $n_x = |N_x|$, where $d \leq n_x \leq 2d$. We say that x is a Δ-*corner* if $n_x(\Delta) = d$ and that it is Δ-*internal* if $n_x(\Delta) = 2d$. Furthermore, x is (Δ, P)-*isolated* if $P_x \neq P_y$ for any $y \in N_x$, while it is (Δ, P)-*generic* if $P_x = P_y$ for any $y \in N_x$.

When using the above notation, we sometimes omit the parameters (e.g. simply writing *isolated* instead of (Δ, P)-isolated) as the context is usually clear.

The definition imposes a symmetric neighborhood relation, that is, $x \in N_y$ holds if and only if $y \in N_x$. If $x \in N_y$ we say that x and y are Δ-neighbours. Note that a point $x = (x_1, \ldots, x_d) \in I$ is a Δ-corner if $x_i < \Delta_i$ or $x_i \geq k - \Delta_i$ for any $1 \leq i \leq d$, and that x is Δ-internal if $\Delta_i \leq x_i < k - \Delta_i$ for any $1 \leq i \leq d$.

▶ **Claim 7.** *Two (Δ, P)-isolated points in I cannot be Δ-neighbors.*

Proof. Suppose towards contradiction that $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ are two distinct (Δ, P)-isolated points and that (x, y) is (i, Δ_i)-related for some $1 \leq i \leq d$. Since $\Delta_i < k/3$, at least one of x or y participates in two different (i, Δ_i)-related pairs: if $x_i < k/3$ then $y_i + \Delta_i = x_i + 2\Delta_i < k$ so y is in two such pairs, and otherwise $x_i \geq \Delta_i$, meaning that x participates in two such pairs. Assume without loss of generality that the two (i, Δ_i)-related pairs are (t, x) and (x, y), then $P_t \neq P_x$ and $P_x \neq P_y$ as x is isolated. By Lemma 4, these are the only (i, Δ_i)-jumps in P.

Choose an arbitrary $j \neq i$ and take $v = (v_1, \ldots, v_d)$ where $v_j = \Delta_j$ and $v_l = 0$ for any $l \neq j$. Recall that $\Delta_j < k/3$, implying that either $x_j + v_j < k$ or $x_j - v_j \geq 0$. Without loss of generality assume the former, and let $x' = x + v$ and $y' = y + v$. Since x and y are (Δ, P)-isolated, and since $x' \in N_x$ and $y' \in N_y$, we get that $P_{x'} \neq P_x \neq P_y \neq P_{y'}$, and thus $P_{x'} \neq P_{y'}$ (as the alphabet is binary). Therefore, (x', y') is also an (i, Δ_i)-jump in P, a contradiction. ◀

▶ **Definition 8.** For three points $x, y, a \in I$, we say that x is *mapped* to y as a result of the flipping of a if $\bar{x} = m(a) + y$. Moreover, define the function $f : I \to I$ as follows: $f(x) = \bar{x} - m(x)$ is the location to which x is mapped as a result of flipping x.

In other words, x is mapped to y as a result of flipping the bit a if bit $\bar{x}$ of A "plays the role" of bit y in the new P-copy Q_a that is created by flipping a. Note that

- If $\bar{x} - m(a) \notin I$ then x is not mapped to any point. However, this cannot hold when $x = a$, so the function f is well defined.

- For a fixed a, the mapping as a result of flipping a is linear: if x and y are mapped to x' and y' respectively, then $y - x = y' - x'$. In particular, if (x, y) is (i, Δ_i)-related for some $1 \leq i \leq d$ then (x', y') is also (i, Δ_i)-related.
- If x is mapped to y as a result of flipping a and $x \neq a$, then $P_x = P_y$.
- On the other hand, we always have $P_x \neq P_{f(x)}$.
- If x is Δ-internal and (Δ, P)-generic, then $f(x)$ must be (Δ, P)-isolated.

The first four statements are easy to verify. To verify the last one, suppose that x is internal and generic and let $z \in N_{f(x)}$; we will show that $P_{f(x)} \neq P_z$. Since x is internal, there exists $y \in N_x$ such that $y - x = z - f(x)$. Then y is mapped to z as a result of flipping x, since $\bar{y} = y + \tilde{k} = z + (x + \tilde{k}) - f(x) = z + \bar{x} - f(x) = z + m(x)$. Therefore $P_y = P_z$. On the other hand, $P_x = P_y$ as x is generic and $P_x \neq P_{f(x)}$, and we conclude that $P_z \neq P_{f(x)}$.

▶ **Lemma 9.** *There is exactly one (Δ, P)-isolated point in I.*

Proof. Let $\mathcal{S}$ be the set of isolated points; our goal is to show that $|\mathcal{S}| = 1$. Consider the set

$$C = \{(x, y) : x, y \in I, (x, y) \text{ is an } (i, \Delta_i)\text{-jump for some } 1 \leq i \leq d\}.$$

Clearly, each point in $\mathcal{S}$ is contained in at least d pairs from C. By claim 7 no pair of isolated points are Δ-neighbours and therefore every pair in C contains at most one point from $\mathcal{S}$. By Lemma 4, $|C| \leq 2d$ which implies that $|\mathcal{S}| \leq 2$. On the other hand we have $|\mathcal{S}| \geq 1$. To see this, observe that the number of (Δ, P)-internal points in I is greater than $\prod_{i=1}^{d} k/3 \geq 2^{d^2}$, while the number of non-Δ-generic points is at most $2|C| \leq 4d$, implying that at least $2^{d^2} - 4d > 0$ of the internal points are generic. Therefore, pick an internal generic point $z \in I$. As we have seen before, $f(z)$ must be isolated.

To complete the proof it remains to rule out the possibility that $|\mathcal{S}| = 2$. If two different (Δ, P)-isolated points $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$ exist, each of them must participate in exactly d pairs in C. This implies that both of them are Δ-corners with d neighbors. It follows that every Δ-internal point z must be generic (since an internal point and a corner point cannot be neighbours), implying that either $f(z) = a$ or $f(z) = b$.

Let $1 \leq i \leq d$ and define $\delta_i > 0$ to be the smallest integer such that there exists an (i, δ_i)-related pair (x, y) of generic internal points with $f(x) = f(y)$. For this choice of x and y we have $m(y) - m(x) = \bar{y} - f(y) - (\bar{x} - f(x)) = \bar{y} - \bar{x} = y - x$, so $(m(x), m(y))$ is also (i, δ_i)-related. In particular, we may take $\Delta_i = \delta_i$ (Recall that until now, we only used the fact that $\Delta_i < k/3$, without committing to a specific value). Without loss of generality we may assume that $f(x) = f(y) = a$. By Claim 5, any pair (s, t) of (i, Δ_i)-related points for which $s \neq \bar{y} - m(y) = f(y) = a$ and $t \neq \bar{x} - m(x) = f(x) = a$ is not an (i, Δ_i)-jump. Since b is not a Δ-neighbour of a, it does not participate in any (i, Δ_i)-jump, contradicting the fact that it is (Δ, P)-isolated. This finishes the proof of the lemma. ◀

Finally, we are ready to show that P is almost homogeneous. Let $a = (a_1, \ldots, a_d)$ be the single (Δ, P)-isolated point in I. Consider the set

$$J = \{x = (x_1, \ldots, x_d) \in I : \Delta_i \leq x_i < \Delta_i + 2^d \text{ for any } 1 \leq i \leq d\}$$

and note that all points in J are Δ-internal. Let $1 \leq i \leq d$ and partition J into $(i, 1)$-related pairs of points. There are $2^{d^2-1} \geq 4d$ pairs in the partition. On the other hand, the number of non-generic points in J is at most $2|C| - (d - 1) < 4d$ (to see it, count the number of elements in pairs from C and recall that a is contained in at least d pairs). Therefore, there exists a pair (x, y) in the above partition such that x and y are both generic. As before,

$f(x)$ and $f(y)$ must be isolated, and thus $f(x) = f(y) = a$, implying that $\Delta_i = \delta_i = 1$. We conclude that $\Delta = (1, \ldots, 1)$.

Claim 5 now implies that any pair (s, t) of $(i, 1)$-related points for which $s \neq \bar{y} - m(y) = f(y) = a$ and $t \neq \bar{x} - m(x) = f(x) = a$ is not an $(i, 1)$-jump. That is, for any two neighbouring points $s, t \neq a$ in I, $P_s = P_t$, implying that $P_x = P_y$ for any $x, y \neq a$ (since $\Delta = (1, \ldots, 1)$, a Δ-neighbour is a neighbour in the usual sense). To see this, observe that for any two points $x, y \neq a$ there exists a path $x_0 x_1 \ldots x_t$ in I where x_j and x_{j+1} are neighbours for any $0 \leq j \leq t-1$, the endpoints are $x_0 = x$ and $x_t = y$, and $x_j \neq a$ for any $0 < j < t$. Since a is isolated, it is also true that $P_a \neq P_x$ for any $x \neq a$.

To finish the proof that P is almost homogeneous, it remains to show that a is a corner. Suppose to the contrary that $0 < a_i < k - 1$ for some $1 \leq i \leq d$ and let $b, c \in I$ be the unique points such that (a, b) and (c, a) are $(i, 1)$-related, respectively. Clearly $f(b) = a$, so a is mapped to $\bar{a} - m(b) = \bar{a} - \bar{b} + f(b) = c - a + a = c$ as a result of flipping b, which is a contradiction - as $P_a \neq P_c$ and $b \neq a, c$. This finishes the proof. ◀

6 From Deletion to Testing

We use the modification lemmas of Section 5 to investigate combinatorial characterizations of the deletion number, which in turn allow efficient approximation and testing of pattern freeness for removable patterns. In particular, we prove that minimal deletion sets and minimal hitting sets are closely related. Due to space considerations, the proofs of all results in this Section do not appear here, and are given in the full version of this paper [9].

The characterizations for almost homogeneous 1-dimensional patterns are also given in the full version of the paper [9], along with an optimal tester for pattern freeness in that case. The rest of this section deals with removable patterns, for both the 1-dimensional and multi-dimensional settings.

In the 1-dimensional case, we show that for any removable pattern there exist certain minimal hitting sets which are in fact minimal deletion sets. These are sets where none of the flips create new occurrences. Our constructive proof shows how to build such a set.

▶ **Theorem 10** ($d_P(S)$ equals $h_P(S)$)**.** *For a binary string S of length n and a binary pattern P of length k that is removable, the deletion number $d_P(S)$ equals the hitting number $h_P(S)$.*

For the multidimensional case, we show that when P is removable, the hitting number $h_P(A)$ of A approximates the deletion number up to a multiplicative constant that depends only on the dimension d. This is done in two stages, the first of which involves the analysis of a procedure that proves the existence of a large collection of P-copies with small pairwise overlaps, among the set of all P-copies in A. This procedure heavily relies on the fact that P is removable. The second stage shows that since these copies have small overlaps, their hitting number cannot be much different than their deletion number. The result is summarized in Lemma 11.

▶ **Lemma 11** (Relation between distance and hitting number)**.** *Let P be a removable (k, d)-array over an alphabet Γ, and let A be an (n, d)-array over Γ. Let $\alpha_d = 4^d + 2^d$. It holds that: $h_P(A) \leq d_P(A) \leq \alpha_d h_P(A) \leq \alpha_d (n/k)^d$.*

We describe efficient distance approximation algorithms and testers for both the 1-dimensional and the d-dimensional removable patterns. The tolerance of the testers depends only on d, and the query complexity is linear in ϵ^{-1} where the constant depends only on d (and not on k; using a completely naive tester, it can be seen that the tolerance and the query

complexity depend on k). The distance approximation algorithms and the testers essentially estimate the hitting number by sampling a small set of $O(k) \times \ldots \times O(k)$ uniformly chosen consecutive subarrays of the input array, and calculating the hitting number of the pattern P in each of these samples.

▶ **Theorem 12** (Approximating the deletion number in 1-dimension). *Let P be a removable string of length k and fix constants $0 < \tau < 1, 0 < \delta < 1/k$. Let $h_1, h_2 : [0,1] \to [0,1]$ be defined as $h_1(\epsilon) = (1-\tau)\epsilon - \delta$ and $h_2(\epsilon) = \epsilon + \delta$. There exists an (h_1, h_2)-distance approximation algorithm for P-freeness with query complexity and running time of $O(1/k\tau\delta^2)$.*

Note that $d_P(S) = h_P(S) \leq n/k$ always holds, so having an additive error parameter of $\delta \geq 1/k$ is pointless. The proof of Theorem 12 can be adapted to derive an (ϵ_1, ϵ_2)-tolerant tester for any $0 \leq \epsilon_1 < \epsilon_2 \leq 1$, yielding the following multiplicative-error tester.

▶ **Corollary 13** (Multiplicative tolerant tester for pattern freeness in 1-dimension). *Fix $0 < \tau < 1$. For any $0 < \epsilon \leq 1$ there exists a $((1-\tau)\epsilon, \epsilon)$-tolerant tester whose number of queries and running time are $O(\epsilon^{-1}\tau^{-3})$.*

For the multidimensional case, our distance approximation algorithm and tolerant tester for P-freeness are given in Theorems 14 and 15.

▶ **Theorem 14** (Approximating the deletion number in multidimensional arrays). *Let P be a removable (k,d)-array and fix constants $0 < \tau \leq 1, 0 \leq \delta \leq 1/k^d$. Let $h_1, h_2 : [0,1] \to [0,1]$ be defined as $h_1(\epsilon) = (1-\tau)^d \alpha_d^{-1}\epsilon - \delta$ and $h_2(\epsilon) = \epsilon + \delta$. There exists an (h_1, h_2)-distance approximation algorithm for P-freeness making at most $\gamma/k^d\tau^d\delta^2$ queries, where $\gamma > 0$ is an absolute constant, and has running time $\zeta_\tau/k^d\delta^2$ where ζ_τ is a constant depending only on τ.*

▶ **Theorem 15** (Multiplicative tolerant tester for pattern freeness in multidimensional arrays). *Fix $0 < \tau \leq 1$ and let P be a removable (k,d)-array. For any $0 < \epsilon \leq 1$ there exists a $((1-\tau)^d \alpha_d^{-1}\epsilon, \epsilon)$-tolerant tester making $C_\tau \epsilon^{-1}$ queries, where $C_\tau = O(1/\tau^d(1-(1-\tau)^d)^2)$. The running time is $C'_\tau \epsilon^{-1}$ where C'_τ depends only on τ.*

7 Discussion and Open Questions

We have provided efficient algorithms for testing whether high-dimensional arrays do not contain P for any fixed removable pattern P. The results suggest several interesting open questions on the problem of pattern-freeness and more generally, on local properties - where we say that a property $\mathcal{P}$ is k-local (for $k \ll n$) if for any array A not satisfying $\mathcal{P}$, there exists a consecutive subarray of A of size at most $k \times \ldots \times k$ which does not satisfy $\mathcal{P}$ as well. That is, a property is local if any array not satisfying $\mathcal{P}$ contains a small 'proof' for this fact. Note that P-freeness is indeed k-local where k is the side length of P, and that a property $\mathcal{P}$ is k-local if and only if there exists a *family* $\mathcal{F}$ of arrays of size at most $k \times \ldots \times k$ each, such that A satisfies $\mathcal{P}$ if and only if it does not contain any consecutive sub-array from $\mathcal{F}$. That is, to understand the general problem of testing local properties of arrays we will need to understand the testing of $\mathcal{F}$-freeness, where $\mathcal{F}$ is a family of forbidden patterns (rather than a single forbidden pattern). As mentioned, it is not clear how to apply our methods to develop testers for families of patterns. Devising testers for this case is an interesting open question.

The problem of approximate pattern matching is of interest as well. The family of forbidden patterns for this problem might consist of a pattern and all patterns that are close enough to it, and the distance measures between patterns might also differ from the Hamming distance (e.g., ℓ_1 distance for grey-scale patterns).

Finally, it is desirable to settle the problem of testing pattern freeness for the almost homogenous case by either finding an efficient tester for the almost homogeneous multi-dimensional case, or proving that an efficient tester cannot exist for such patterns. It is also of interest to examine which of the $[k]^d$ patterns with $k < 3 \cdot 2^d$ are removable.

Acknowledgements. We are grateful to Swastik Kopparty for numerous useful comments. We are thankful to Sofya Raskhodnikova for her useful feedback.

References

1. N. Alon (2002). Testing subgraphs in large graphs, *Random structures and algorithms*, 21(3–4):359–370.
2. N. Alon, O. Ben-Eliezer and E. Fischer (2017). *Testing hereditary properties of ordered graphs and matrices,* arXiv preprint 1704.02367.
3. N. Alon, M. Krivelevich, I. Newman and M. Szegedy (2001). Regular languages are testable with a constant number of queries, *SIAM Journal on Computing*, 30, 1842–1862.
4. N. Alon, E. Fischer, M. Krivelevich and M. Szegedy (2000). Efficient testing of large graphs, *Combinatorica*, 20, 451–476.
5. N. Alon, E. Fischer and I. Newman (2007). Efficient testing of bipartite graphs for forbidden induced subgraphs, *SIAM Journal on Computing*, 37.3, 959–976.
6. N. Alon and J. Spencer (2008). *The Probablistic Method.* Wiley.
7. A. Amir, G. Benson (1998). Two-Dimensional Periodicity in Rectangular Arrays, *SIAM Journal on Computing*, 27, 90–106.
8. A. Amir, G. Benson, M. Farach (1994). An Alphabet Independent Approach to Two-Dimensional Pattern Matching, *SIAM Journal on Computing*, 23, 313–323.
9. O. Ben-Eliezer, S. Korman and D. Reichman (2017). *Deleting and Testing Forbidden Patterns in Multi-Dimensional Arrays,* arXiv preprint 1607.03961.
10. P. Berman, M. Murzabulatov, S. Raskhodnikova (2015). Constant-Time Testing and Learning of Image Properties, arXiv prepreint 1503.01363.
11. P. Berman, M. Murzabulatov and Sofya Raskhodnikova (2016). Tolerant Testers of Image Properties. *ICALP*, 1–90:14.
12. R.S. Boyer and J.S. Moore (1977). A fast string searching algorithm, *Comm. ACM*, 20(10), 762–772.
13. R. Cole (1991). Tight Bounds on the Complexity of the Boyer-Moore String Matching Algorithm, *SODA*, 224–233.
14. C. Maxime, A. Czumaj, L. Gasieniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter (1994). Speeding up two string-matching algorithms, *Algorithmica*, 12, 247–267.
15. E. Fischer, and I. Newman (2001). Testing of matrix properties, *STOC*, 286–295
16. E. Fischer and I. Newman (2007). Testing of matrix-poset properties, *Combinatorica*, 27(3), 293–327.
17. E. Fischer, E. Rozenberg, Lower bounds for testing forbidden induced substructures in bipartite-graph-like combinatorial objects, Proc. RANDOM 2007, 464–478.
18. Z. Galil, J. G. Park and K. Park. (2004). Three-dimensional periodicity and its application to pattern matching. *SIAM Journal on Discrete Mathematics*, 18(2), 362–381.
19. Z. Galil and J. I. Seiferas (1983). Time-Space-Optimal String Matching, *J. Comput. Syst. Sci*, 26(3), 280–294.
20. O. Goldreich, S. Goldwasser and D. Ron (1998). Property testing and its connection to learning and approximation, *JACM*, 45, 653–750.
21. J. Kärkkäinen and E. Ukkonen (2008). Multidimensional string matching. In *Encyclopedia of Algorithms*, 559–562.

22 I. Kleiner, D. Keren, I. Newman, O. Ben-Zwi (2011). Applying Property Testing to an Image Partitioning Problem, *IEEE Trans. Pattern Anal. Mach. Intell*, 33(2), 256–265.

23 S. Korman, D. Reichman, G. Tsur and S. Avidan. Fast-Match: Fast Affine Template Matching, International Journal of Computer Vision, to appear.

24 D. E. Knuth, J. H. Morris Jr. and V. R. Pratt (1977). Fast Pattern Matching in Strings, *SIAM J. Comput.* 6(2): 323–350.

25 T. Lecroq (2007). Fast exact string matching algorithms, *Information Processing Letters*, 102(6), 229–235.

26 G. Navarro and M. Raffinot (2000). Fast and flexible string matching by combining bit-parallelism and suffix automata, *Journal of Experimental Algorithmics (JEA)* 5:4.

27 M. Parnas, D. Ron and R. Rubinfeld (2006). Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6), 1012–1042.

28 S. Raskhodnikova (2003). Approximate testing of visual properties, *RANDOM*, 370–381.

29 R. L. Rivest (1977). On the Worst-Case Behavior of String-Searching Algorithms, *SIAM J. Comput.* 6(4): 669–674.

30 R. Rubinfeld and M. Sudan (1996). Robust characterization of polynomials with applications to program testing, *SIAM J. Comput.* 25, 252–271.

31 G. Tsur and D. Ron (2014). Testing properties of sparse images, *ACM Transactions on Algorithms* 4.

32 A. C. Yao (1977). Probabilistic computation, towards a unified measure of complexity, *FOCS*, 222–227.

On the Value of Penalties in Time-Inconsistent Planning*†

Susanne Albers[1] and Dennis Kraft[2]

1 **Department of Computer Science, Technical University of Munich, Munich, Germany**
albers@in.tum.de

2 **Department of Computer Science, Technical University of Munich, Munich, Germany**
kraftd@in.tum.de

Abstract

People tend to behave inconsistently over time due to an inherent present bias. As this may impair performance, social and economic settings need to be adapted accordingly. Common tools to reduce the impact of time-inconsistent behavior are penalties and prohibition. Such tools are called commitment devices. In recent work Kleinberg and Oren [5] connect the design of a prohibition-based commitment device to a combinatorial problem in which edges are removed from a task graph G with n nodes. However, this problem is NP-hard to approximate within a ratio less than $\sqrt{n}/3$ [2]. To address this issue, we propose a penalty-based commitment device that does not delete edges, but raises their cost. The benefits of our approach are twofold. On the conceptual side, we show that penalties are up to $1/\beta$ times more efficient than prohibition, where $\beta \in (0, 1]$ parameterizes the present bias. On the computational side, we improve approximability by presenting a 2-approximation algorithm for allocating penalties. To complement this result, we prove that optimal penalties are NP-hard to approximate within a ratio of 1.08192.

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

Keywords and phrases approximation algorithms, behavioral economics, commitment devices, computational complexity, time-inconsistent preferences

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.10

1 Introduction

Most people make long term plans. They intend to eat healthy, save money, prepare for exams, exercise regularly and so on. Curiously, the same people often change their plans at a later point in time. They indulge in fast food, squander their money, fail to study and skip workouts. Although change may be necessary due to unforeseen events, people often change their plans even if the circumstances stay the same. This type of *time-inconsistent behavior* is a well-known phenomenon in behavioral economics and might impair a person's performance in social or economic domains [1, 8].

A sensible explanation for time-inconsistent behavior is that people are *present biased* and assign disproportionately greater value to the present than to the future. Consider, for instance, a scenario in which a student named Alice attends a course over several weeks. To pass the course, Alice either needs to solve a homework exercise each week or give a

* The full version can be found at arXiv:1702.01677 [cs.DS], https://arxiv.org/abs/1702.01677.
† Work supported by the ERC, Grant Agreement No. 691672.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 10; pp. 10:1–10:12


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

presentation once. The presentation incurs a onetime effort of 3, whereas each homework exercise incurs an effort of 1. Furthermore, we assume that Alice immediately fails if she misses a homework assignment before she has given a presentation. If the course lasts for more than 3 weeks, she clearly minimizes her effort by giving a presentation in the first week. Paradoxically, if Alice is present biased, she may solve all homework exercises instead. The reason for this is the following:

Suppose Alice perceives present effort accurately, but discounts future effort by a factor of $\beta = 1/3$. In the first week Alice must decide between solving the homework exercise or giving a presentation. Remember that she automatically fails if she does neither of the two. Clearly, the homework incurs less immediate effort than the presentation. Furthermore, Alice can still give a presentation later on. Her perceived effort for doing the homework this week and giving the presentation the week after is $1 + \beta 3 = 2$. To Alice this plan appears more convenient than giving the presentation right away. Consequently, she does the homework. However, come next week she changes this plan and postpones the presentation once more. Her reasoning is the same as in the first week. Due to her time-inconsistent behavior, Alice continues to postpone the presentation and ends up doing all the homework assignments.

Previous Work. Time-inconsistent behavior has been studied extensively in behavioral economics. For an introduction to the topic refer for example to [1]. Alice's scenario demonstrates how time-inconsistency arises whenever people are present biased. Alice evaluates her preferences based on a well-established discounting model called *quasi-hyperbolic-discounting* [7]. As her story shows, quasi-hyperbolic-discounting tempts people to make poor decisions. To prevent poor decisions, social and economic settings need to be adapted accordingly. Depending on the domain, such adaptations might be implemented by governments, companies, teachers or people themselves. We call these entities *designers* and their motivation can be benevolent or self-serving. In either case, the designer's objective is to commit people to a certain goal. Their tools are called *commitment devices* and may include rewards, penalty fees and strict prohibition [3, 9].

Until recently, the study of time-inconsistent behavior lacked a unifying and expressive framework. However, groundbreaking work by Kleinberg and Oren closed this gap by reducing the behavior of a quasi-hyperbolic-discounting person to a simple planning problem in task graphs [5]. Their framework has helped to identify various structural properties of social and economic settings that affect the performance of present biased individuals [5, 10]. It has also been extended to people whose present bias varies over time [4] as well as people who are aware of their present bias and act accordingly [6]. We will formally introduce the framework in Section 2. A significant part of Kleinberg and Oren's work is concerned with the study of a simple yet powerful commitment device based on prohibition [5]. In particular, they demonstrate how performance can be improved by removing a strategically chosen set of edges from the task graph. The drawback of their approach is its computational complexity. As it turns out, an optimal commitment device is NP-hard to approximate within a ratio less than $\sqrt{n}/3$, where n denotes the number nodes in the task graph [2]. Currently, the only known polynomial-time algorithms achieve approximation ratios of $1/\beta$ and $1 + \beta/n$, which in combination yields a $1 + \sqrt{n}$ approximation [2]. It should be mentioned that Kleinberg and Oren's framework has also been used to analyze reward-based commitment devices [2, 10]. Unfortunately, the computational complexity of such commitment devices does not permit a polynomial-time approximation within a finite ratio unless P = NP [2].

Our Contribution. To circumvent the theoretical limitations mentioned above, we propose a natural generalization of Kleinberg and Oren's work. Instead of prohibition, our commitment

device is based on penalty fees, a standard tool in the economic literature [3, 9]. This means that the designer is free to raise the cost of arbitrary edges in the task graph. We call such an assignment of penalties a *cost configuration*. The designer's objective is to construct cost configurations that are as efficient as possible.

In Section 3 we conduct a quantitative comparison between the efficiency of penalty fees and prohibition. Assuming that optimal commitment devices can be computed, we show that penalties are strictly more powerful than prohibition. In particular, we show that penalties may outperform prohibition by a factor of $1/\beta$ where β parameterizes the present bias. This result is tight. In Section 4 we investigate the computational complexity of our commitment device. Using a reduction from 3-SAT, we argue that the construction of an efficient cost configuration is NP-hard when posed as a decision problem. A generalization of this reduction proves NP-hardness for approximations within a ratio of 1.08192. Unless P = NP, this dismisses the existence of a polynomial-time approximation scheme. While analyzing the complexity of our commitment device we also point to a remarkable structural property. More specifically, we show that every cost configuration admits another cost configuration of comparable efficiency that assigns its cost entirely along a single path. Assuming that the path is known in advance, we provide an algorithm for constructing such a cost configuration in polynomial-time. This result is important for the design of exact algorithms as it reduces the search space to the set of paths through the task graph. Finally, Section 5 introduces a 2-approximation algorithm for our commitment device. This is our main result and a considerable improvement to the $\sqrt{n}/3$ approximability of the prohibition-based commitment device [2]. It is interesting to note that the $1/\beta$ approximation algorithm of [2] can also be applied to penalty fees. As a result, our penalty-based approach is particularly interesting for highly present biased agents with $\beta < 1/2$.

2 The Formal Framework

In the following, we introduce Kleinberg and Oren's framework [5]. Let $G = (V, E)$ be a directed acyclic graph with n nodes that models a given long-term project. The edges of G correspond to the tasks of the project and the nodes represent the states. In particular, there exists a start state s and a target state t. Each path from s to t corresponds to a valid sequence of tasks to complete the project. The effort of a specific task is captured by a non-negative cost $c(e)$ assigned to the associated edge e.

To complete the project, an agent with a *present bias* of $\beta \in (0, 1]$ incrementally constructs a path from s to t as follows: At any node v different from t, the agent evaluates her *lowest perceived cost*. For this purpose she considers all paths P leading from v to t. However, she only anticipates the cost of the first edge of P correctly; all other edges of P are discounted by her present bias. More formally, let $d(w)$ denote the cost of a cheapest path from node w to t. The agent's lowest perceived cost at v is defined as $\zeta(v) = \min\{c(v, w) + \beta d(w) \mid (v, w) \in E\}$. We assume that she only traverses edges (v, w) that minimize her anticipated cost, i.e. edges for which $c(v, w) + \beta d(w) = \zeta(v)$. Ties are broken arbitrarily. For convenience, we define the perceived cost of (v, w) as $\eta(v, w) = c(v, w) + \beta d(w)$. The agent is motivated by an intrinsic or extrinsic reward r collected at t. As she receives this reward in the future, she perceives its value as βr at each node different from t. When located at v, she compares her lowest perceived cost to the anticipated reward and continues moving if and only if $\zeta(v) \leq \beta r$. Otherwise, if $\zeta(v) > \beta r$, we assume she abandons the project. We call G *motivating* if she does not abandon while constructing her path from s to t. Note that in some graphs the agent can take several paths from s to t due to ties between incident edges. In this case, G is considered motivating if she does not abandon on any of these paths.

For the sake of a clear presentation, we will assume throughout this work that each node of G is located on a path from s to t. This assumption is sensible for the following reason: Clearly, the agent can only visit nodes that are reachable from s. Furthermore, she is not willing to enter nodes that do not lead to the reward. Consequently, only nodes that are on a path from s to t are relevant to her behavior. Note that all nodes that do not satisfy this property can be removed from G in a simple preprocessing step.

To illustrate the model, we revisit Alice's scenario from Section 1. Assume that the course takes m weeks. We represent each week i by a distinct node v_i and set $s = v_1$. Furthermore, we introduce a target node t that marks the passing of the course. Each week $i < m$ Alice can either give a presentation or proceed with the homework. We model the first case by an edge (v_i, t) of cost 3 and the latter case by an edge (v_i, v_{i+1}) of cost 1. In the last week, i.e. $i = m$, Alice's only sensible choice is to do the homework. Therefore, edge (v_m, t) is of cost 1. Recall that Alice's present bias is $\beta = 1/3$. Moreover, assume that her intrinsic reward for passing is $r = 6$. For $i < m$ her perceived cost of the edges (v_i, t) is $\eta(v_i, t) = c(v_i, t) = 3$. As this is less than her perceived reward, which is $\beta 6 = 2$, she is never motivated to give a presentation right away. However, her perceived cost of the edges (v_i, v_{i+1}) is at most $\eta(v_i, v_{i+1}) \leq c(v_i, v_{i+1}) + \beta c(v_{i+1}, t) \leq 2$. This matches her perceived reward. As a result, she walks from v_1 to v_m along the edge (v_i, v_{i+1}). Once she reaches v_m she traverses the only remaining edge for a perceived cost of $\eta(v_m, t) = c(v_m, t) = 1$ and passes the course. This matches our analysis from Section 1.

3 Prohibition versus Penalties

In this section we demonstrate how the *designer* can modify a given project to help the agent reach t. For this purpose, the designer may have several commitment devices at her disposal. A straightforward approach is to increase the reward that the agent collects at t. Although this may keep the agent from abandoning the project prematurely, it has no influence on the path taken by the agent. Furthermore, increasing the reward may be costly for the designer. As a result, the designer has two conflicting objectives. On the one hand, she must ensure that the agent reaches t. On the other hand, she needs to minimize the resources spent. To deal with this dilemma, Kleinberg and Oren allow the designer to prohibit a strategically chosen set of tasks [5]. This commitment device is readily implemented in their framework. In fact, it is sufficient to remove all edges of prohibited tasks. The result is a subgraph G' that may significantly reduce the reward required to motivate the agent. Unfortunately, an optimal subgraph G' is NP-hard to approximate within a ratio less than $\sqrt{n}/3$ [2].

To circumvent this theoretical limitation, we propose a different approach. Instead of prohibiting certain tasks we allow the designer to charge penalty fees. Such fees could be implemented in several ways; for instance in the form of donations to charity. Our only assumption is that the designer does not benefit from the fees, i.e. there is no incentive to maximize the fees payed by the agent. Similar to the prohibition-based commitment device, our commitment device is readily implemented in Kleinberg and Oren's framework. The designer simply assigns a positive extra cost $\tilde{c}(e)$ to the desired edges e. The new cost of e is equal to $c(e) + \tilde{c}(e)$. We call $\tilde{c}$ a *cost configuration*. Applying a cost configuration to G yields a new task graph with increased edge cost. All concepts of the original framework carry over immediately. Sometimes it will be necessary to compare different commitment devices with each other. To clarify which commitment device we are talking about, we use the following notation whenever necessary: If we consider a subgraph G', we write $d_{G'}$, $\eta_{G'}$ and $\zeta_{G'}$. Similarly, if we consider a cost configuration $\tilde{c}$, we write $d_{\tilde{c}}$, $\eta_{\tilde{c}}$ and $\zeta_{\tilde{c}}$. Moreover, we denote the trivial cost configuration, i.e. the one that assigns no extra cost, by $\tilde{0}$.

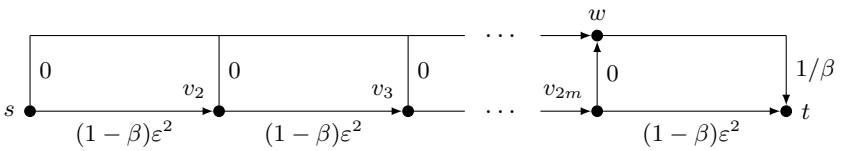


Figure 1 Graph maximizing the ratio between the efficiency of subgraphs and cost configurations.

It is interesting to think of penalty fees as a natural generalization of prohibition. This becomes particularly apparent in the context of Kleinberg and Oren's framework as we can recreate the properties of any subgraph G' by a cost configuration $\tilde{c}$. For this purpose, it is sufficient to assign an extra cost of $\tilde{c}(e) = r + 1$ to any edge e not contained in G'. As a result, the agent's perceived cost of paths along e certainly exceeds her perceived reward. However, this means that e is irrelevant to the agent's planning and could be deleted from G altogether. Consequently, penalties are at least as powerful as prohibition. But how much more efficient are penalties in the best case? As the following theorem suggests, cost configurations may outperform subgraphs by a factor of almost $1/\beta$.

▶ **Theorem 1.** *The ratio between the minimum reward r that admits a motivating subgraph and the reward q of a motivating cost configuration is at most $1/\beta$. This bound is tight.*

Proof. To see that $r/q \leq 1/\beta$, let G be an arbitrary task graph and consider a subgraph G' whose only edges are those of a cheapest path P from s to t. Recall that $d(s)$ denotes the cost of P. In G' the agent's only choice is to follow P. Because her perceived cost is a discounted version of the actual cost, she never perceives a cost greater than $d(s)$ in G'. Consequently, $d(s)/\beta$ is an upper bound on r. Next, consider an arbitrary cost configuration $\tilde{c}$. As $\tilde{c}$ only increases edge cost, the agent's lowest perceived cost at s is at least $\beta d(s)$. We conclude that q must be at least $d(s)$ to be motivating. This yields the desired ratio.

It remains to show the tightness of the result. For this purpose, we construct a task graph G such that: (a) The minimum reward that admits a motivating subgraph is $1/\beta^2$. (b) There exists a cost configuration that is motivating for a reward of $(1+\varepsilon)/\beta$, where ε is a positive value strictly less than 1. Our construction is a modified version of Alice's task graph. Let $m = \lceil \beta^{-2}(1-\beta)^{-1}\varepsilon^{-2} \rceil$ and assume that G contains a path $v_1, \ldots, v_{2m+1}$ whose edges are all of cost $(1-\beta)\varepsilon^2$. We call this the *main path* and set $s = v_1$ and $t = v_{2m+1}$. In addition to the main path, each v_i with $i \leq 2m$ has a *shortcut* to t via a common node w. The edges (v_i, w) are free, whereas (w, t) is of cost $1/\beta$. Figure 1 illustrates the structure of G. Note that the drawing merges some of the edges (v_i, w) for a concise representation.

We proceed to argue that G satisfies (a). For the sake of contradiction, assume the existence of a subgraph G' that is motivating for a reward $r < 1/\beta^2$. In this case the agent must not take shortcuts as her perceived cost at w exceeds her perceived reward. Therefore, she must follow the main path. In particular, she must visit each node v_i on the first half of the path, i.e. $i \leq m+1$. At each of these nodes, her lowest perceived cost is realized along the edge (v_i, v_{i+1}). Essentially, there are two ways she can come up with this cost. First, she might plan to take a shortcut at a later point in time. As a result, we get $\eta_{G'}(v_i, v_{i+1}) \geq c(v_i, v_{i+1}) + \beta c(w, t) > 1$. Secondly, she might plan to stay on the main path. In this case she must traverse at least m edges, each of which contributes $\beta(1-\beta)\varepsilon^2$ or more to $\eta_{G'}(v_i, v_{i+1})$. Consequently, we get $\eta_{G'}(v_i, v_{i+1}) \geq m\beta(1-\beta)\varepsilon^2 \geq 1/\beta \geq 1$. Either way her perceived cost for taking the main path is at least 1. As this tempts her to take the shortcut at v_i, all of the first $m+1$ shortcuts must be interrupted in G'. This means she must walk along at least m edges of the main path before taking the first shortcut. As a

Algorithm 1: PathAndFence

Input: Task graph G, present bias β, path $P = v_1, \dots, v_m$, positive value ε
Output: Cost configuration $\tilde{c}$

```
1 c̃ ← 0̃;
2 for i from m − 1 to 1 do
3     foreach w ∈ {w′ | (v_i, w′) ∈ E} do
4         if w ≠ v_{i+1} then c̃(v_i, w) ← max{0, η_c̃(v_i, v_{i+1}) − η_c̃(v_i, w) + βε/(m − 2)};
5 return c̃;
```

result, her lowest perceived cost at v_1 is at least $\zeta_{G'}(v_1) \geq m\beta(1-\beta)\varepsilon^2 \geq 1/\beta$. This is a contradiction to the assumption that r is motivating.

Next we show how to construct a cost configuration $\tilde{c}$ that satisfies (b). For this purpose it is sufficient to add an extra cost of ε to all edges (v_i, w). To upper bound the agent's perceived cost of (v_i, v_{i+1}), assume she plans to take a shortcut in the next step, i.e. at v_{i+1}. For $i < 2m$ we get $\eta_{\tilde{c}}(v_i, v_{i+1}) \leq c(v_i, v_{i+1}) + \beta(\tilde{c}(v_{i+1}, w) + c(w, t)) = (1-\beta)\varepsilon^2 + \beta\varepsilon + 1 < 1 + \varepsilon$. In the special case of $i = 2m$, the inequality $\eta_{\tilde{c}}(v_i, v_{i+1}) < 1 + \varepsilon$ is still satisfied, this time via the direct edge (v_{2m}, t). In contrast, the agent's perceived cost of an immediate shortcut is $\eta_{\tilde{c}}(v_i, t) = \varepsilon + \beta c(w, t) = 1 + \varepsilon$ for all $i \leq 2m$. Therefore, she is never tempted to divert from the main path. Furthermore, a reward of $q = (1+\varepsilon)/\beta$ is sufficient to keep her motivated. ◀

4 Computing Motivating Cost Configurations

We now turn our attention to the computational aspects of designing efficient penalty fees. In this section, we assume that the agent's reward is fixed to some value $r > 0$. Our goal is to compute cost configurations that are motivating for r whenever they exist. Similar to the prohibition-based commitment device [2], this task is NP-hard whenever the agent is present biased, i.e. $\beta \neq 1$. We will prove this claim at the end of the section. But first, assume that we already have partial knowledge of the solution. More precisely, assume we know one of the paths the agent might take in a motivating cost configuration provided a motivating cost configuration exists. We call this path P. Based on P, Algorithm 1 constructs a cost configuration $\tilde{c}$ that is motivating for a slightly larger reward $r + \varepsilon$.

The basic idea of Algorithm 1 is simple. Starting with v_{m-1}, it considers all nodes v_i of P in reverse order. For each v_i it assigns an extra cost of $\max\{0, \eta_{\tilde{c}}(v_i, v_{i+1}) - \eta_{\tilde{c}}(v_i, w) + \beta\varepsilon/(m-2)\}$ to the edges (v_i, w) that leave P, i.e. edges different from (v_i, v_{i+1}). As a result, the agent's perceived cost of (v_i, w) is greater than that of (v_i, v_{i+1}) by at least $\beta\varepsilon/(m-2)$. Consequently, she has no incentive to divert from P at v_i. Since the algorithm runs in reverse order, extra cost assigned in iteration i has no effect on the agent's behavior at later nodes, i.e. nodes v_j with $j > i$. Figuratively speaking, the algorithm builds a fence of penalty fees along P preventing the agent from leaving P. For this reason, we call the algorithm PathAndFence. As the next proposition suggests, cost configurations of this particular fence structure can achieve almost the same efficiency as any other cost configuration. Due to space constraints, refer to the full version of this work for a proof.

▶ **Proposition 2.** *Let P be the agent's path from s to t with respect to a cost configuration $\tilde{c}^*$ that is motivating for a reward r.* PathAndFence *constructs a cost configuration $\tilde{c}$ that is motivating for a reward of $r + \varepsilon$, where ε is an arbitrary small but positive quantity.*

Proposition 2 has some interesting implications. The first one is of conceptual nature.

Note that PathAndFence constructs a cost configuration that never actually charges the agent any extra cost. This suggests the existence of an efficient penalty-based commitment device that does not require the designer to enforce penalties. The mere threat of repercussions appears to be sufficient. The second implication is computational. Clearly, PathAndFence runs in polynomial-time with respect to n. In particular, the number of iterations does not depend on the choice of ε. Consequently, PathAndFence can be combined with an exhaustive search algorithm that considers all paths from s to t to search for a motivating cost configuration. Although the number of such paths can be exponential in n, this approach still reduces the size of the search space considerably. Finally, it should be noted that a similar result for the prohibition-based commitment device is unlikely to exist. The reason is that subgraphs remain hard to approximate even if the agent's optimal path is known [2], indicating a favorable computational complexity for the design of penalty fees. Of course there is another potential source of hardness: the computation of P. To prove that this is a limiting factor, we introduce the decision problem MOTIVATING COST CONFIGURATION:

▶ **Definition 3** (MCC)**.** Given a task graph G, a reward $r > 0$ and a present bias $\beta \in (0, 1]$, decide the existence of a motivating cost configuration.

We propose a reduction from 3-SAT to show that MCC is NP-complete for arbitrary $\beta \in (0, 1)$. At a later point we will use the same reduction to establish a hardness of approximation result.

▶ **Theorem 4.** *MCC is NP-complete for any present bias $\beta \in (0, 1)$.*

Proof. According to [2], whether or not a given task graph is motivating for a fixed reward can be verified in polynomial-time. Of course, this remains valid if the edges are assigned extra cost. Consequently, any motivating cost configuration is a suitable certificate for a "yes"-instance of MCC. We conclude that MCC is in NP. In the following, we present a reduction from 3-SAT to show that MCC is also NP-hard. This establishes the theorem.

Let $\mathcal{I}$ be an arbitrary instance of 3-SAT consisting of ℓ clauses $c_1, \ldots, c_\ell$ over m variables $x_1, \ldots, x_m$. We construct a MCC instance $\mathcal{J}$ such that its task graph G admits a motivating cost configuration for a reward of $r = 1/\beta$ if and only if $\mathcal{I}$ has a satisfying variable assignment. Figure 2 depicts G for a small sample instance of $\mathcal{I}$. In general, G consists of a source s, a target t and five nodes $u_1, \ldots, u_5$. Depending on $\mathcal{I}$, G also contains some extra nodes. For each variable x_k, there are two *variable nodes* $w_{k,T}$ and $w_{k,F}$. The idea is to interpret x_k as true whenever the agent visits $w_{k,T}$ and as false whenever she visits $w_{k,F}$. As a result, the agent's walk through G yields a variable assignment τ. Furthermore, for each clause c_i there is a *literal node* $v_{i,j}$ corresponding to the j-th literal of c_i. Our goal is to construct G in such a way that every motivating cost configuration guides the agent along literal nodes that are satisfied with respect to τ.

All nodes $v_{i,j}$ and $w_{k,y}$ are connected via so-called *forward edges*. More specifically, for all $1 \leq i < \ell$ and $1 \leq j, j' \leq 3$ there is a forward edge from $v_{i,j}$ to $v_{i+1,j'}$. Similarly, there is a forward edge from $w_{k,y}$ to $w_{k+1,y'}$ for all $1 \leq k < m$ and $y, y' \in \{T, F\}$. We also have forward edges from s to each $v_{1,j}$, from each $v_{\ell,j}$ to u_1, from u_2 to each $w_{1,y}$ and from each $w_{m,y}$ to u_3. For the sake of readability, some forward edges are merged in Figure 2. The price of each forward edge is $(1-\beta)^3 - \varepsilon$, where the encoding length of β is assumed to be polynomial in $\mathcal{I}$. Furthermore, ε denotes a small but positive quantity such that

$$\varepsilon < \min\left\{(1-\beta)^2, \frac{\beta(1-\beta)^3}{1+\beta}, \frac{\beta(1-\beta)^2}{1+\beta}\right\}.$$

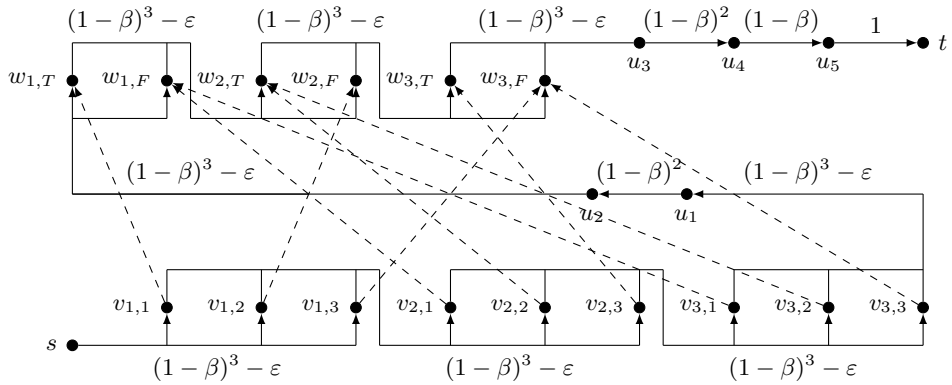


Figure 2 Reduction from the 3-SAT instance: $(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$.

In addition to the forward edges, there are three types of *shortcuts*. The first type, which is depicted as dashed edges in Figure 2, connects each literal node $v_{i,j}$ to a distinct variable node via a single edge of cost $(1-\beta)^2$. If the j-th literal of c_i is equal to x_k, the shortcut goes to $w_{k,F}$. Otherwise, if the literal is negated, i.e. $\bar{x}_k$, the shortcut goes to $w_{k,T}$. The second type of shortcut goes from u_2 to t along a single edge of cost $2-\beta$. For clear representation, this shortcut is omitted in Figure 2. The third type of shortcut connects each variable node $w_{k,y}$ to t via a distinct intermediate node. The first edge is free while the second costs $2-\beta$. Again, shortcuts of this type are omitted in Figure 2 to keep the drawing simple. Finally, there are four more edges (u_1, u_2), (u_3, u_4), (u_4, u_5) and (u_5, t) of cost $(1-\beta)^2$, $(1-\beta)^2$, $1-\beta$ and 1 respectively. Note that G is acyclic and its encoding length is polynomial in $\mathcal{I}$.

To establish the theorem, we must show that $\mathcal{I}$ has a satisfying variable assignment if and only if $\mathcal{J}$ has a motivating cost configuration. A detailed argument is described in the full version of this work. At this point we only sketch the main ideas. For this purpose let $\tilde{c}$ be a cost configuration that is motivating for a reward of $1/\beta$ and let P be the agent's path through G with respect to $\tilde{c}$. Note that P cannot contain shortcuts of the second or third type as their edges are too expensive. Furthermore, P cannot contain a shortcut of the first type because the agent either perceives it as too expensive or is tempted to enter a shortcut of the third type immediately afterwards. As a result, P contains exactly one of the two nodes $w_{k,T}$ and $w_{k,F}$ for each variable x_k. Let $\tau : \{x_1, \dots, x_m\} \to \{T, F\}$ be the corresponding variable assignment. To keep the agent on P, $\tilde{c}$ must assign extra cost to all shortcuts that start at a variable node satisfied by τ. However, this raises the perceived cost of all paths via literal nodes not satisfied by τ to values that are not motivating. Consequently, P cannot contain such literal nodes. But P must contain exactly one literal node of each clause because P takes no shortcuts. This means that τ satisfies at least one literal in each clause and is therefore a feasible solution of $\mathcal{I}$. Conversely, whenever $\mathcal{I}$ has a feasible solution τ, we can construct a motivating cost configuration $\tilde{c}$ as follows: First, assign an appropriate extra cost, e.g. $(1-\beta)^2$, to the shortcuts of type three starting at the variable nodes $w_{k,\tau(x_k)}$. Secondly, block the forward edges into the variable nodes $w_{k,\tau(\bar{x}_k)}$ with high extra cost of e.g. 1. ◀

5 Approximating Motivating Cost Configurations

The previous section suggests that an optimal assignment of penalty fees is NP-hard to compute. This section therefore focuses on an optimization version of the problem. Our goal is to construct cost configurations that require the designer to raise the reward at t as little

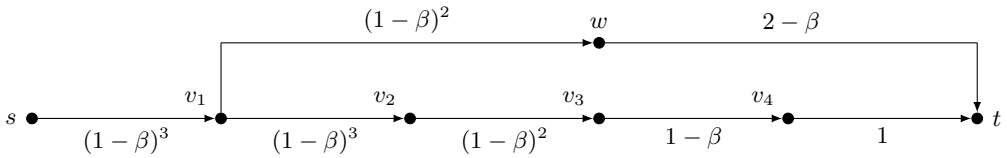


Figure 3 Task graph with no optimal cost configuration.

Algorithm 2: MinMaxPathApprox

Input: Task graph G, present bias β
Output: Cost configuration $\tilde{c}$

```
P ← minmax path from s to t with respect to η_{c̃0};
ϱ ← max{η_{c̃0}(e) | e ∈ P};
foreach v ∈ V \ {t} do
    ς(v) ← successor node of v on a cheapest path from v to t;
foreach (v, w) ∈ E do
    if (v, w) ∈ P ∨ (ς(v) = w ∧ v ∉ P) then c̃(v, w) ← 0;
    else if (v, w) ≠ P ∧ ς(v) ≠ w then c̃(v, w) ← 3ϱ/β;
    else
        P' ← v, ς(v), ς(ς(v)), ..., t;
        u ← first node of P' different from v that is also a node of P;
        e ← most expensive edge of P', between v and u;
        c̃(v, w) ← c(e);
return c̃;
```

as possible. However, before we provide a formal definition of the problem we should consider a curious technical detail; namely, not all task graphs admit an optimal cost configuration.

Consider, for instance, the task graph in Figure 3. At v_1 the agent is indifferent between the edges (v_1, v_2) and (v_1, w). In both cases her perceived cost is 1. If she chooses (v_1, w), she faces a perceived cost of $2 - \beta$ at w. Conversely, if she chooses (v_1, v_2), she perceives a cost of 1 at v_2, v_3 and v_4. Assuming that $\beta < 1$, (v_1, v_2) is the better choice. To break the tie between (v_1, w) and (v_1, v_2) we must place a positive extra cost of ε onto the upper path. However, when located at s the agent's perceived cost of the upper path is $1 + \beta\varepsilon$. In contrast, her perceived cost of the lower path is $1 + \beta(1-\beta)^3$. Assuming that $\varepsilon < (1-\beta)^3$, she prefers the upper path. Consequently, we can construct a cost configuration that is motivating for a reward arbitrarily close to $1/\beta$, but no cost configuration is motivating for a reward of exactly $1/\beta$. To account for the potential lack of an optimal solution, we compare our results to the infimum of all rewards that admit a motivating cost configuration. The optimization problem MCC-OPT is defined accordingly:

▶ **Definition 5** (MCC-OPT). Given a task graph G and a present bias $\beta \in (0, 1)$, determine the infimum of all rewards for which a motivating cost configuration exists.

We are now ready to introduce Algorithm 2. This algorithm enables us to construct cost configurations that approximate MCC-OPT within a factor of 2. At a high level, the algorithm proceeds in two phases. First, it computes a value ϱ such that ϱ/β is a lower bound for any reward that admits a motivating cost configuration. Secondly, it constructs a cost configuration $\tilde{c}$ that is motivating for a reward of $2\varrho/\beta$. This yields the promised approximation ratio of 2.

For a more detailed discussion of Algorithm 2 assume that each edge e is labeled with its perceived cost $\eta_{\tilde{0}}(e)$. Furthermore, let $\tilde{c}'$ be an arbitrary cost configuration and P' the agent's corresponding path from s to t. Our goal is to lower bound the minimum reward that is motivating for $\tilde{c}'$ by some value ϱ/β. For this purpose, it is instructive to observe that any motivating reward must be at least $\max\{\eta_{\tilde{c}'}(e) \mid e \in P'\}/\beta \geq \max\{\eta_{\tilde{0}}(e) \mid e \in P'\}/\beta$. Since P' can be an arbitrary path from s to t, we set

$$\varrho = \min\big\{\max\{\eta_{\tilde{0}}(e) \mid e \in P\} \;\big|\; P \text{ is a path from } s \text{ to } t\big\}.$$

In other words, ϱ is the maximum edge cost of a *minmax path* P from s to t with respect to $\eta_{\tilde{0}}$. Note that P can be computed in polynomial-time by adding the edges of G in non-decreasing order of perceived cost to an initially empty set E' until s and t become connected for the first time. Any path from s to t that only uses edges of E' is a suitable minmax path.

We continue with the construction of $\tilde{c}$. To facilitate this task, Algorithm 2 sets up a cheapest path successor relation ς. More precisely, it assigns a distinct successor node $\varsigma(v)$ to each $v \in V \setminus \{t\}$. The successor is chosen in such a way that $(v, \varsigma(v))$ is the initial edge of a cheapest path from v to t. Since we may assume that t is reachable from each node of G, all $v \neq t$ must have at least one suitable successor. By construction of ς, any path $P' = v, \varsigma(v), \varsigma(\varsigma(v)), \ldots, t$ is a cheapest path from v to t. We call P' the *ς-path* of v.

Once ς has been created, Algorithm 2 starts to assign an appropriate extra cost to all edges of G. The idea behind this assignment is to either keep the agent on P or guide her along a suitable ς-path. For this reason, we also call the algorithm MinMaxPathApprox. While iterating through the edges (v, w) of G the algorithm distinguishes between three types of edges: First, (v, w) might be an edge of P or an edge of a ς-path. In the latter case v must not be a node of P. Any (v, w) that satisfies these requirements is an edge we want the agent to traverse or use in her plans. Consequently, (v, w) is assigned no extra cost. Secondly, (v, w) might neither be an edge of P nor of a ς-path. Since we do not want the agent to traverse or plan along such an edge, the algorithm assigns an extra cost of $3\varrho/\beta$ to (v, w). This is sufficiently expensive for the agent to lose interest in (v, w) provided that the reward is $2\varrho/\beta$. Thirdly, (v, w) might not be an edge of P but of a ς-path such that v is a node of P. This is the most involved case. To find an appropriate cost for (v, w), the algorithm considers the ς-path P' of v. Let u be the first common node between P and P' that is different from v. Because P and P' both end in t, such a node must exist. Moreover, let e be the most expensive edge of P' between v and u. The algorithm assigns an extra cost of $c(e)$ to (v, w). As we will show in Theorem 6, this cost is either high enough to keep the agent on P or she travels to u along P' without encountering edges that are too expensive.

Clearly, Algorithm 2 can be implemented to run in polynomial-time with respect to the size of G. It remains to show that the algorithm returns a cost configuration $\tilde{c}$ that approximates MCC-OPT within a factor of 2.

▶ **Theorem 6.** MinMaxPathApprox *has an approximation ratio of* 2*.*

Proof. Recall that ϱ denotes the maximum perceived edge cost along the minmax path P. From the above description of MinMaxPathApprox, it should be evident that ϱ/β is a lower bound on the minimum motivating reward of any cost configuration. To prove the theorem, we need to show that the algorithm returns a cost configuration $\tilde{c}$ that is motivating for a reward of $2\varrho/\beta$.

As our first step we argue that the cost of a cheapest path from any node v to t with respect to $\tilde{c}$ is at most twice the cost of a cheapest path with respect to $\tilde{0}$. More formally, we prove that $d_{\tilde{c}}(v) \leq 2d_{\tilde{0}}(v)$. For this purpose let P' be the ς-path of v. By construction

of ς, P' is a cheapest path from v to t. It is crucial to observe that MinMaxApprox only assigns extra cost to an edge $(v', \varsigma(v'))$ of P' if v' is located on P. Consequently, there is at most one edge with extra cost between any two consecutive intersections of P and P'. Furthermore, this extra cost is equal to the cost of an edge on P' between v' and the next intersection of P and P'. Therefore, each edge of P' can contribute at most once to the total extra cost assigned to P'. This means that the price of P' with respect to $\tilde{c}$ is at most twice the price of P' with respect to $\tilde{0}$. Because the price of P' is an upper bound for $d_{\tilde{c}}(v)$, we have shown that $d_{\tilde{c}}(v) \leq 2d_{\tilde{0}}(v)$.

We proceed to investigate the agent's walk through G. Our goal is to show that her lowest perceived cost is at most 2ϱ at every node v on her way. This establishes the theorem. Our analysis is based on the following case distinction: First, assume that v is located on P. The immediate successor of v on P is denoted by w. Remember that $\tilde{c}$ assigns no extra cost to (v, w). Using the result from the previous paragraph, we get

$$\begin{aligned}\zeta_{\tilde{c}}(v) &\leq \eta_{\tilde{c}}(v,w) = c(v,w) + \beta d_{\tilde{c}}(w) \leq c(v,w) + \beta 2 d_{\tilde{0}}(w) \leq 2\big(c(v,w) + \beta d_{\tilde{0}}(w)\big) \\ &= 2\eta_{\tilde{0}}(v,w) \leq 2\varrho.\end{aligned}$$

The last inequality is valid by definition of ϱ.

Secondly, assume that v is not located on P and consider the last node v' on P the agent visited before v. Because she traversed $(v', \varsigma(v'))$ to get to v, we know that $\eta_{\tilde{c}}(v', \varsigma(v')) \leq 2\varrho$ and $d_{\tilde{c}}(\varsigma(v')) \leq 2\varrho/\beta$. We also know that she faces an extra cost of $3\varrho/\beta$ whenever she tries to leave the ς-path P' of v' before the next intersection of P and P'. Since she is not willing to pay this much, v must be located on P'. In particular, all paths from $\varsigma(v')$ to t either visit $\varsigma(v)$ or cross an edge that charges an extra cost of $3\varrho/\beta$. Consequently, a cheapest path from $\varsigma(v')$ to t with respect to $\tilde{c}$ costs at least $d_{\tilde{c}}(\varsigma(v')) \geq \min\{3\varrho/\beta, d_{\tilde{c}}(\varsigma(v))\}$. As $d_{\tilde{c}}(\varsigma(v')) \leq 2\varrho/\beta$, this implies that $d_{\tilde{c}}(\varsigma(v')) \geq d_{\tilde{c}}(\varsigma(v))$. Our proof is almost complete. For the final part, recall that $(v, \varsigma(v))$ is located on P' between v' and the next intersection of P and P'. By construction of $\tilde{c}$ we have $\tilde{c}(v', \varsigma(v')) \geq c(v, \varsigma(v))$. Furthermore, $(v, \varsigma(v))$ has no extra cost. Putting all the pieces together we get

$$\begin{aligned}\zeta_{\tilde{c}}(v) &\leq \eta_{\tilde{c}}(v, \varsigma(v)) = c(v, \varsigma(v)) + \beta d_{\tilde{c}}(\varsigma(v)) \leq \tilde{c}(v', \varsigma(v')) + \beta d_{\tilde{c}}(\varsigma(v')) \\ &\leq c(v', \varsigma(v')) + \tilde{c}(v', \varsigma(v')) + \beta d_{\tilde{c}}(\varsigma(v')) = \eta_{\tilde{c}}(v', \varsigma(v')) \leq 2\varrho.\end{aligned}$$

◀

To complement this result, we argue that MCC-OPT is NP-hard to approximate within any ratio of $1 + \beta(1-\beta)^4$ or less. Choosing $\beta = 1/5$ maximizes this term and yields inapproximability for constant ratios of 1.08192 or less. In particular, assuming that P $\neq$ NP this rules out the existence of a polynomial-time approximation scheme.

▶ **Theorem 7.** *MCC-OPT is NP-hard to approximate within a ratio less or equal to* 1.08192.

Proof. To establish the theorem, a reduction similar to the one from Theorem 4 can be used. In fact, given a 3-SAT instance $\mathcal{I}$ we can construct the corresponding MCC-OPT instance $\mathcal{J}$ the same way as in the proof of Theorem 4. The only difference is that our choice of ε is slightly more restrictive as we require

$$\varepsilon < \min\Big\{\beta(1-\beta)^3, \beta(1-\beta)^2(2-\beta), \frac{\beta^2(1-\beta)^3}{1+\beta}, \frac{\beta^2(1-\beta)^2(2-\beta)}{1+\beta}\Big\}.$$

The proof can be structured around the following properties of $\mathcal{J}$: (a) If $\mathcal{I}$ has a solution, $\mathcal{J}$ admits a motivating cost configuration for a reward of $1/\beta$. (b) If $\mathcal{I}$ has no solution, $\mathcal{J}$ admits no motivating cost configuration for a reward of $(1+\beta(1-\beta)^4)/\beta$ or less. Consequently,

any algorithm that approximates MCC-OPT within a ratio of $1+\beta(1-\beta)^4$ or less must also solve $\mathcal{I}$. To maximize this ratio we choose $\beta = 1/5$ and obtain the desired approximability bound, namely $1 + (1 - 1/5)^4/5 = 1.08192$. All that remains to show is that $\mathcal{J}$ indeed satisfies (a) and (b). The correctness of (a) is an immediate consequence of the proof of Theorem 4. A detailed proof of (b) can be found in the full version of this work. ◀

6 Conclusion

In this work we have used Kleinberg and Oren's graph theoretic framework [5] to provide a systematic analysis of a penalty-based commitment device. We have shown that penalty fees are strictly more powerful than prohibition. In particular, we have shown that penalties may outperform prohibition by a factor of up to $1/\beta$. We have also been able to obtain some of the first positive computational results for the algorithmic design of commitment devices. We have given a polynomial-time algorithm to construct penalty fees that match an optimal solution by a factor of 2. This is significant progress when compared to the prohibition-based commitment device, whose approximation is known to be NP-hard within a factor less than $\sqrt{n}/3$ [2]. Due to its versatility, expressiveness and favorable computational properties, we believe that our penalty-based commitment device will prove to be a valuable tool for addressing time-inconsistent behavior in complex social and economic settings.

References

1 George A. Akerlof. Procrastination and obedience. *The American Economic Review*, 81(2):1–19, 1991.

2 Susanne Albers and Dennis Kraft. Motivating time-inconsistent agents: A computational approach. In *Proceedings of the 12th Conference on Web and Internet Economics*, pages 309–323. Springer, 2016.

3 Gharad Bryan, Dean Karlan, and Scott Nelson. Commitment devices. *Annual Review of Economics*, 2:671–698, 2010.

4 Nick Gravin, Nicole Immorlica, Brendan Lucier, and Emmanouil Pountourakis. Procrastination with variable present bias. In *Proceedings of the 17th ACM Conference on Economics and Computation*, pages 361–361, New York, NY, USA, 2016. ACM.

5 Jon Kleinberg and Sigal Oren. Time-inconsistent planning: A computational problem in behavioral economics. In *Proceedings of the 15th ACM Conference on Economics and Computation*, pages 547–564, New York, NY, USA, 2014. ACM.

6 Jon Kleinberg, Sigal Oren, and Manish Raghavan. Planning problems for sophisticated agents with present bias. In *Proceedings of the 17th ACM Conference on Economics and Computation*, pages 343–360, New York, NY, USA, 2016. ACM.

7 David Laibson. Golden eggs and hyperbolic discounting. *The Quarterly Journal of Economics*, pages 443–477, 1997.

8 Ted O'Donoghue and Matthew Rabin. Doing it now or later. *The American Economic Review*, 89:103–124, 1999.

9 Ted O'Donoghue and Matthew Rabin. Incentives and self control. *Advances in Economics and Econometrics: The 9th World Congress*, 2:215–245, 2006.

10 Pingzhong Tang, Yifeng Teng, Zihe Wang, Shenke Xiao, and Yichong Xu. Computational issues in time-inconsistent planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017. To appear.

Efficient Approximations for the Online Dispersion Problem*†

Jing Chen[1], Bo Li[2], and Yingkai Li[3]

1 **Department of Computer Science, Stony Brook University, Stony Brook, NY, USA**
jingchen@cs.stonybrook.edu
2 **Department of Computer Science, Stony Brook University, Stony Brook, NY, USA**
boli2@cs.stonybrook.edu
3 **Department of Computer Science, Stony Brook University, Stony Brook, NY, USA**
yingkli@cs.stonybrook.edu

Abstract

The *dispersion* problem has been widely studied in computational geometry and facility location, and is closely related to the packing problem. The goal is to locate n points (e.g., facilities or persons) in a k-dimensional polytope, so that *they are far away from each other and from the boundary of the polytope.* In many real-world scenarios however, the points arrive and depart at different times, and decisions must be made without knowing future events. Therefore we study, for the first time in the literature, the *online dispersion* problem in Euclidean space.

There are two natural objectives when time is involved: the *all-time worst-case* (ATWC) problem tries to maximize the minimum distance that ever appears at any time; and the *cumulative distance* (CD) problem tries to maximize the integral of the minimum distance throughout the whole time interval. Interestingly, the online problems are highly non-trivial even on a segment. For cumulative distance, this remains the case even when the problem is time-dependent but offline, with all the arriving and departure times given in advance.

For the online ATWC problem on a segment, we construct a deterministic polynomial-time algorithm which is $(2\ln 2+\epsilon)$-competitive, where $\epsilon > 0$ can be arbitrarily small and the algorithm's running time is polynomial in $\frac{1}{\epsilon}$. We show this algorithm is actually *optimal*. For the same problem in a square, we provide a 1.591-competitive algorithm and a 1.183 lower-bound. Furthermore, for arbitrary k-dimensional polytopes with $k \geq 2$, we provide a $\frac{2}{1-\epsilon}$-competitive algorithm and a $\frac{7}{6}$ lower-bound. All our lower-bounds come from the structure of the online problems and hold even when computational complexity is not a concern. Interestingly, for the offline CD problem in arbitrary k-dimensional polytopes, we provide a polynomial-time black-box reduction to the online ATWC problem, and the resulting competitive ratio increases by a factor of at most 2. Our techniques also apply to online dispersion problems with different boundary conditions.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases dispersion, online algorithms, geometric optimization, packing, competitive algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.11

* A full version of this extended abstract is available at http://arxiv.org/abs/1704.06823.

† The authors thank Joseph Mitchell for motivating us to study the online dispersion problem. We thank Esther Arkin, Michael Bender, Rezaul A. Chowdhury, Jie Gao, Joseph Mitchell, Jelani Nelson, and the participants of the Algorithm Reading Group for helpful discussions, and several anonymous reviewers for helpful comments.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 11; pp. 11:1–11:15

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

1 Introduction

The problem of assigning elements to locations in a given area comes up only too often in real life: where to seat the customers in a restaurant, where to put certain facilities in a city, where to build nuclear power stations in a country, etc. Different problems have different features and constraints, but one common feature that appears in many of them is *not to locate the elements too close to each other*: for people's privacy, for environmental safety, and/or for serving more users. Another feature is also common in many applications: that is, *not to locate the elements too close to the boundary of the area.* Indeed, for security reasons, important national industrial facilities in many countries are built at a safe distance away from the border. Such problems have been widely studied in computational geometry and facility location; see, e.g., [32, 3, 2, 4]. In particular, in the *dispersion* problem defined by [2], there is a k-dimensional polytope P and an integer n, and the goal is to locate n points in P so as to maximize *the minimum distance among them and from them to the boundary of* P.

However, there is another important feature in all the scenarios mentioned above and many other real-world scenarios: the presence of elements is *time-dependent* and decisions need to be made along time, without knowing when the elements will come and go in the future. Indeed, it may be hard to move an element once it is located, making it infeasible for the decision maker to relocate all the present elements according to the optimal static solution when an arrival/departure event occurs. Online dispersion and facility location problems have been studied when the underlying locations are vertices of a graph [28, 17, 27]. In this paper we consider, for the first time in the literature, the online dispersion problem in Euclidean space. The arriving and departure times of points are chosen by an adversary who knows everything and works adaptively. An online dispersion algorithm decides where to locate a point upon its arrival, without any knowledge about future events.

1.1 Main Results

We focus on two natural objectives for the online problem: the *all-time worst-case* (ATWC) problem, which aims at maximizing the minimum distance that ever appears at any time; and the *cumulative distance* (CD) problem, which aims at maximizing the integral of the minimum distance throughout the whole time interval. Although polynomial-time constant approximations have been given when time is not involved [2, 4], nothing was known about the online problem. As we will show, solutions for the online problem are already complex even on a segment. For cumulative distance, even when the problem is time-dependent but offline, with all the arriving and departure times given in advance, it remains unclear how to efficiently compute the optimal solution. We formally define the problem in Section 2 and summarize our results in Table 1 below.

The most technical parts are the online ATWC problem and the offline time-dependent CD problem. Interestingly, we provide an efficient reduction from the offline CD problem to the online ATWC problem, and show that in order to solve the former, one can use an algorithm for the latter as a black-box. For the online ATWC problem, it is not hard to see that a natural greedy algorithm provides a 2-competitive ratio. Our main contributions for this problem are to provide an efficient algorithm that is optimal for the 1-dimensional case, improve the competitive ratio and prove a lower-bound for squares, and provide an efficient implementation of the greedy algorithm for the general case. We also prove a simple lower-bound for the general case.

To establish our results, we show interesting new connections between dispersion and ball-packing – both *uniform* packing (i.e., with balls of identical radius) and *non-uniform*

Table 1 Online and offline time-dependent dispersion problems in a k-dimensional polytope P.

	Online	Offline time-dependent
ATWC	$k = 1$: a $2\ln 2$ (≈ 1.386) lower-bound and an optimal algorithm; see Theorems 5 and 8. $k = 2$: a 1.183 lower-bound and a 1.591-competitive algorithm for squares; see Theorems 11 and 12. $k \geq 2$: a $\frac{7}{6}$ lower-bound and a $\frac{2}{1-\epsilon}$-competitive algorithm for arbitrary polytopes P; see Theorems 13 and 14.	Equivalent to dispersion without time; see Claim 2.
CD	No constant competitive algorithm even when $k = 1$; see Claim 1.	A black-box reduction to online ATWC for arbitrary k and P, with the competitive ratio scaling up by at most 2; see Theorem 15.

packing (i.e., with balls of different radii). All our algorithms are deterministic and of polynomial time. Some of them take an arbitrarily small constant ϵ as a parameter and the running time is polynomial in $\frac{1}{\epsilon}$. All inapproximability results hold even when running time is not a concern. Due to lack of space, most proofs are given in the full version [8].

Discussion and future directions. An algorithm for high-dimensional polytopes may not be directly applicable in dimension 1, because all locations are on the boundary when a segment is treated as a high-dimensional polytope, and the minimum distance is always 0. Accordingly, we do not know whether the lower-bound for dimension 1 carries through to higher dimensions, and proving better lower-bounds will be an interesting problem for future studies. In [8], we consider online dispersion without the boundary constraint, where the lower-bound for dimension 1 indeed carries through. We show all our algorithms can be adapted for this setting. Another important future direction is to understand the role of randomized algorithms in the online dispersion problem. Finally, improving the (deterministic or randomized) algorithms' competitive ratios in various classes of polytopes is certainly a long-lasting theme for the online dispersion problem. Special classes such as regular polytopes and uniform polytopes may be reasonable starting points. Given the connections between dispersion and ball-packing, it is conceivable that new competitive algorithms for online dispersion may stem from and also imply new findings on ball-packing.

1.2 Related Work

Dispersion without time. In dispersion problems in general, the possible locations can be either a *continuous* region or a set of *discrete* candidates. Two objectives have been studied in the literature: the *max-min distance* as considered in this paper, and the *maximum total distance*. In continuous settings, the authors of [2] consider the max-min distance with the boundary condition. Under L_∞-norm, they give a polynomial-time 1.5-approximation in rectilinear polygons[1] and show that a $\frac{14}{13}$-approximation in arbitrary polygons is NP-hard. Moreover, they show there is no PTAS under any norm unless P=NP. [4] considers a similar boundary condition under L_2-norm and provides a 1.5-approximation in polygons with

[1] A rectilinear polygon is a 2-dimensional polytope whose edges are axis-parallel.

obstacles. [11] considers the problem of selecting n points in n given unit-disks, one per disk, and the objective is to maximize the minimum distance.

In discrete settings, [32, 5] show that, if the distances among the candidate locations do not satisfy triangle inequality, then there is no polynomial-time constant approximation for either objective, unless P=NP; while if triangle inequality is satisfied, then there are efficient 2-approximations for both objectives. If the goal is to maximize total distance and the candidate locations are in a k-dimensional space, [15] gives a PTAS under L_1-norm; and [6, 7] provide PTASes when locations need to satisfy matroid constraints. Finally, [3, 31, 25, 1] consider various dispersion problems in obnoxious facility allocation.

Packing without time. It is well known that dispersion and packing are "dual" problems of each other [26]. In this paper we show interesting new connections between them and use several important results for packing in our analysis. Thus we briefly introduce this literature. Indeed, the packing problem is one of the most extensively studied problems in geometric optimization, and a huge amount of work has been done on different variants of the problem; see [30, 21] for surveys on this topic.

One important problem is to *pack circles with identical radius, as many as possible, in a bounded region.* [18] shows this problem to be strongly NP-hard and [22] gives a PTAS for it. An APTAS for the *circle bin packing* problem is given in [29]. The *dispersal packing* problem tries to maximize the radius of a given number of circles packed in a square. A lot of effort has been made in finding the optimal radius and the corresponding packing when the number of circles is a small constant; see [36, 35, 37, 30]. Heuristic methods have also been used in finding approximations when the number of circles gets large [24, 41]. Finally, an important packing problem is to understand the *packing density*: that is, the maximum fraction of an infinite space covered by a packing of unit circles/spheres. The packing density is solved for dimension 2 in [14] and for dimension 3 in [20]. Very recently, [42] and [9] solve it for dimensions 8 and 24, respectively. Asymptotic lower bounds (as the dimension grows) for the density of the densest packing are provided in [33].

Online geometric optimization. Many important geometric optimization problems have been studied in online settings, although the settings and objectives are quite different from ours. In particular, the seminal work of [38] provides a nearly-optimal competitive algorithm for the classic *online bin-packing* problem. Algorithms for variants of the problem have been considered ever since, such as a constant competitive ratio for packing circles in square bins [23], and constant competitive ratios for bin-packing in higher dimensions [12, 13].

In *online facility location* [28], it is the demands rather than the facilities that come along time. The facilities have open costs and the goal is to minimize the total open cost and the total distance between demands and facilities. As shown in [28], when the demands arrive adversarially, there is a randomized polynomial-time $O(\log n)$-competitive algorithm, and a constant competitive ratio is impossible. A deterministic $O(\frac{\log n}{\log \log n})$-competitive algorithm and a matching lower-bound are provided in [17]. In *incremental facility location* [16], the facilities can be opened, closed or merged, depending on the arriving demands. In [27, 34], there is a cost for each location configuration and the goal is to minimize the cost when the facilities arrive online. A constant competitive algorithm for this problem is provided in [27], and [34] gives a reduction from the online problem to the offline version of the problem.

Dynamic resource division. Fair resource division is an important problem in economics [39, 10, 40]. When the resource is 1-dimensional and homogenous, dynamic fair division

is in some sense the "dual" of online dispersion: locating n points as far as possible from each other and from the boundary is the same as partitioning the segment into $n+1$ pieces as evenly as possible. [19] provides an optimal d-disruptive mechanism for 1-dimensional homogenous resource. Interestingly, our algorithm for the 1-dimensional case provides an optimal mechanism when $d = 1$, although the techniques are quite different. Optimal mechanisms for heterogenous or high-dimensional resource remain unknown. It would be interesting to see if our techniques for dispersion can be used in resource division in general.

2 The Online Dispersion Problem

Given a k-dimensional polytope P, the *dispersion* problem [2] takes as input a positive integer n and outputs n locations, $X_1, \ldots, X_n \in P$, for n points. For each point i, let $dis(X_i, \partial P)$ be the distance from X_i to ∂P, the boundary of P, measured by L_2-norm. Also, let $dis(X_i, X_j)$ be the distance between X_i and X_j for any $i \neq j$. The objective is

$$Disp(n; P) \triangleq \max_{X_1,\ldots,X_n \in P} \min_{i,j\in[n]} \{dis(X_i, \partial P), dis(X_i, X_j)\}.$$

In [8], we also consider the dispersion problem where the distances to the boundary are not taken into consideration. Most of our techniques can be applied there.

We now define the *online dispersion* problem, where each point i arrives at time s_i and departs at time d_i, with $d_i > s_i$. Without loss of generality, $0 = s_1 \leq s_2 \leq \cdots \leq s_n$. An online algorithm is notified upon an arrival/departure event. It must decide the location X_i for i upon its arrival, knowing neither the future events nor the number n. An adversary knows how the algorithm works and chooses future events after seeing the algorithm's output so far. In the *time-dependent offline* version of the problem, the times of all events, denoted by a vector $S = ((s_1, d_1), \ldots, (s_n, d_n))$, is given to the algorithm in advance.

Given such a vector S, let $T = \max_{i\in[n]} d_i$ be the last departure time. Moreover, given locations $X = (X_1, \ldots, X_n)$, for any $t \leq T$, let

$$d_{min}(t; X) = \min_{i,j\in[n]: s_i \leq t \leq d_i, s_j \leq t \leq d_i} \{dis(X_i, \partial P), dis(X_i, X_j)\}$$

be the minimum distance corresponding to the points that are present at time t. When X is clear from the context, we may write $d_{min}(t)$ for short. We consider two natural objectives: the *all-time worst-case* (ATWC) problem, where the objective is

$$OPT_A(S; P) \triangleq \max_{X_1,\ldots,X_n} \min_{t\leq T} d_{min}(t);$$

and the *cumulative distance* (CD) problem, where the objective is

$$OPT_C(S; P) \triangleq \max_{X_1,\ldots,X_n} \int_0^T d_{min}(t)dt.$$

Note that both objectives are defined to be the optimum of the corresponding *offline* problems, the same as the *ex-post* optimum for the online problems. Below we provide two simple observations about the objectives.

▶ **Claim 1.** *For the CD problem, even when $k = 1$ and P is the unit segment, no (randomized) online algorithm achieves a competitive ratio to OPT_C better than $\Omega(n)$.*

Next, given any ex-post instance $S = ((s_1, d_1), \ldots, (s_n, d_n))$, let m be the maximum number of points simultaneously present at any time t: that is, $m = \max_{t\leq T} |\{i : s_i \leq t \leq d_i\}|$.

▶ **Claim 2.** *$\forall\ S = ((s_1, d_1), \ldots, (s_n, d_n))$, letting $m = \max_{t \le T} |\{i : s_i \le t \le d_i\}|$, we have $OPT_A(S; P) = Disp(m; P)$.*

In light of the claims above, the online CD problem is highly inapproximable and the offline ATWC problem is equivalent to the dispersion problem without time. Thus we will focus on the online ATWC problem and the offline CD problem, especially the former. Our results imply a simple $O(n)$-competitive algorithm for the online CD problem (see [8]), matching the lower-bound in Claim 1.

Below we point out some connections between dispersion and ball-packing: they are not hard to show, and similar results for the dispersion problem without the boundary condition have been pointed out in [26]. More precisely, the *(uniform) ball-packing* problem [22] in a polytope P takes as input a non-negative value r and outputs an integer n, the maximum number of balls of radius r that can be packed non-overlappingly in P, together with a corresponding packing. We denote the solution by $Pack(r; P)$. The *dispersal packing* problem [2] is a "mixture" of dispersion and packing: it takes as input an integer n and outputs the maximum radius for n balls with identical radius that can be packed in P, together with a corresponding packing. That is, $DP(n; P) \triangleq \max\{r : Pack(r; P) \ge n\}$.

Recall that a k-dimensional convex polytope P has an *insphere* if the largest ball contained wholly in P is tangent to *all* the facets (i.e., $(k-1)$-faces) of P. Such a ball, if it exists, is unique. It is referred to as *the* insphere of P. The center of the insphere maximizes the minimum distance for any point in P to its facets, and has the same distance to all facets – the radius of the insphere. We have the following two claims.

▶ **Claim 3.** *For any $k \ge 1$ and any k-dimensional convex polytope P with an insphere, letting x be the radius of the insphere, we have $Disp(n; P) = \frac{2xDP(n;P)}{x+DP(n;P)}$.*

▶ **Claim 4.** *For any $k \ge 1$ and any k-dimensional convex polytope P with an insphere, given the radius of the insphere,*
(1) *any polynomial-time algorithm for $Disp(n; P)$ implies such an algorithm for $Pack(r; P)$;*
(2) *any polynomial-time algorithm for $Pack(r; P)$ implies an FPTAS for $Disp(n; P)$.*

To the best of our knowledge, it is still unknown whether ball-packing in regular polytopes (which is a special case of convex polytopes with an insphere) is NP-hard or not. Therefore the complexity of dispersion in regular polytopes remains open. Note that ball-packing in arbitrary polytopes is NP-hard [18], so is a $\frac{14}{13}$-approximation for dispersion in rectilinear polygons [2]. Moreover, a claim similar to Claim 4 applies to $DP(n; P)$ and $Pack(r; P)$ in arbitrary polytopes. The relation between dispersion and packing in arbitrary polytopes is not so clear and worth further investigation: for example, it would be interesting to know if there exists a counterpart of Claim 4 when the polytope does not have an insphere.

Finally, the *insert-only* model, where all points have the same departure time, is a special case of our general model. Interestingly, as will become clear in our analysis, the difficulty of the general online ATWC problem is captured by the problem under this special model. The insert-only model was also considered by [27, 34] in settings different from ours and with a different objective function. We further discuss this model in [8].

3 The 1-Dimensional Online All-Time Worst-Case Problem

Note that a 1-dimensional polytope is simply a segment. Without loss of generality, we consider the unit segment $P = [0, 1]$. Below we first provide a lower bound for the competitive ratio of any algorithm, even computationally unbounded ones.

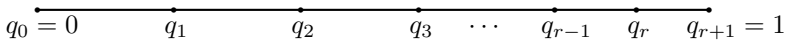


Figure 1 The pre-fixed positions in Q for dimension 1.

3.1 The Lower Bound

▶ **Theorem 5.** *No online algorithm achieves a competitive ratio better than* $2\ln 2$ *(*≈ 1.386*) for the 1-dimensional ATWC problem.*

Proof Ideas. Letting $\sigma'_r = \sum_{i=r+1}^{2r} \frac{1}{i}$ for any positive integer r, we show that no algorithm achieves a competitive ratio better than $2\sigma'_r$. Roughly speaking, we construct an instance (i.e., an adversary) for the online ATWC problem with three stages. In the first stage, $r-1$ points arrive simultaneously; in the second stage, r new points arrive one by one; and finally, all $2r-1$ points depart simultaneously. If an algorithm $\mathcal{A}$ is α-competitive to OPT_A with $\alpha < 2\sigma'_r$, it must be α-competitive after the arrival of each point, as it does not know the total number of points. Thus for each arriving point, there must exist an interval long enough such that putting the new point inside the interval does not violate the competitive ratio. We show that in order for $\mathcal{A}$ to do so, the segment must be longer than P itself, a contradiction. Theorem 5 holds by setting $r \to \infty$. The complete proof is in [8]. ◀

3.2 A Polynomial-Time Online Algorithm

Next, we provide a deterministic polynomial-time online algorithm whose competitive ratio to OPT_A can be arbitrarily close to $2\ln 2$. Intuitively, a good algorithm should disperse the points as evenly as possible. However, if at some point of time with m points present, the resulting $m+1$ intervals on the segment have almost the same length, then the next arriving point will force the minimum distance to drop by a factor of 2, while the optimum only changes from $\frac{1}{m+1}$ to $\frac{1}{m+2}$, causing the competitive ratio to drop by almost 2. To overcome this problem, the algorithm must find a balance between two consecutive points, choosing a sub-optimal solution for the former so as to leave enough space for the latter. The difficulty, as for online algorithms in general, is that this balance needs to be kept for arbitrarily many pairs of consecutive point, as the sequence of points is chosen by an adversary who observes the algorithm's output. Inspired by our lower bound, roughly speaking, our algorithm uses a parameter r to pre-fix the locations of the first r points and the resulting $r+1$ intervals, and then inserts the next $r+1$ points in the middle of these intervals. The idea is that, when done properly, after these $2r+1$ points, the resulting configuration is almost the same as if the algorithm has used parameter $2r+1$ to pre-fix the first $2r+2$ intervals: then the procedure can repeat for arbitrary sequences.

More specifically, given a positive integer r, let $Q = \{q_1, \ldots, q_r\}$ be a set of positions on the segment, such that the length ratios of the $r+1$ intervals sliced by them are $\frac{1}{r+1} : \frac{1}{r+2} : \cdots : \frac{1}{2r+1}$. That is, letting $\sigma_r = \sum_{i=r+1}^{2r+1} \frac{1}{i}$, the lengths of the intervals are $\frac{1}{\sigma_r(r+1)}$, $\frac{1}{\sigma_r(r+2)}$, $\cdots$, $\frac{1}{\sigma_r(2r+1)}$, and $q_i = \frac{1}{\sigma_r} \sum_{j=r+1}^{r+i} \frac{1}{j}$ for each $i \in [r]$, as illustrated by Figure 1, with $q_0 = 0$ and $q_{r+1} = 1$. Note that σ_r differs from σ'_r in Theorem 5 by $\frac{1}{2r+1}$. Also, σ_r is strictly decreasing in r and $\lim_{r\to\infty} \sigma_r = \ln 2$. Moreover, for any two intervals (q_{j-1}, q_j) and $(q_{j'-1}, q_{j'})$ with $j < j' \le r+1$, we have $|(q_{j'-1}, q_{j'})| < |(q_{j-1}, q_j)| < 2|(q_{j'-1}, q_{j'})|$.

Our algorithm, Algorithm 1, also takes as parameter an ordering for the positions in Q, denoted by $\tau = (\tau_1, \tau_2, \ldots, \tau_r)$. We have the following two lemmas, whose main ideas are sketched below. Recall that, given $S = ((s_1, d_1), \ldots, (s_n, d_n))$, m is the maximum number of points simultaneously present at any time t.

Algorithm 1. A polynomial-time algorithm for the 1-dimensional online ATWC problem.

Parameter: A positive integer r, the corresponding set $Q = \{q_1, \ldots, q_r\}$, and an ordering τ for Q.

Input: A sequence of points arriving and departing along time.

Denote by $\hat{Q}$ the set of positions ever occupied by a point. At any point of time, a position in $\hat{Q}$ is labeled *occupied* if currently there is a point there and *vacant* otherwise. Initially $\hat{Q} = \emptyset$.
When a point i leaves, change the label of its position in $\hat{Q}$ from *occupied* to *vacant*.
When a point i arrives:
if $\hat{Q} = \emptyset$ or all positions in $\hat{Q}$ are labelled *occupied* **then**
  if $Q \not\subseteq \hat{Q}$ **then**
    Choose the first position q according to τ with $q \in Q \setminus \hat{Q}$, add it to $\hat{Q}$ and label it *occupied*.
    Put i at position q.
  else
    Find position q which is the middle of the largest interval created by the positions in $\hat{Q}$.
    Put i at position q, add q to $\hat{Q}$ and label it *occupied*.
  end if
else
  Arbitrarily choose a vacant position q from $\hat{Q}$ and label it *occupied*.
  Put i at position q.
end if

▶ **Lemma 6.** *$\forall$ r and τ, Algorithm 1 is $2\sigma_r$-competitive to OPT_A for any S with $m > r$.*

Proof Ideas. Since Algorithm 1 only creates a new position when the number of points simultaneously present on the line increases, for any time t, the number of positions created is exactly the maximum number of points that has appeared simultaneously on the line. Thus only m positions is created for instance S. We prove that when $m > r$, the minimum distance produced by our algorithm, denoted by $d_{min}(m)$, is $\frac{1}{2^{l+1}\sigma_r(r+i)}$, where l, i are the unique integers such that $l \geq 0, 0 \leq i \leq r+1$ and $2^l(r+1) + 2^l(i-1) \leq m < 2^l(r+1) + 2^l i$. Note that the minimum distance only depends on m. By comparing OPT_A with $d_{min}(m)$, we show that the competitive ratio $2\sigma_r$ holds for all $m > r$. ◀

▶ **Lemma 7.** *For any integer $l > 0$ and $r = 2^l - 1$, there exists an ordering τ for the corresponding set Q, s.t. Algorithm 1 is $2\sigma_r$-competitive to OPT_A for any S with $m \leq r$.*

Proof Ideas. Interestingly, due to the structure of Algorithm 1, we only need to consider the instance $S = ((1, r+1), (2, r+1), \ldots, (r, r+1))$. We construct an ordering $\tau = \{\tau_d\}_{d \in [r]}$ for Q such that the competitive ratio at any time $d \in [r]$ is smaller than $2\sigma_r$. To do so, we fill in a complete binary tree with r nodes as in Figure 2, and τ is obtained by traversing the tree in a breadth-first manner starting from the root. Given any $d = 2^i + s$ with $i \in \{0, 1, \ldots, l-1\}$ and $s \in \{0, 1, \ldots, 2^i - 1\}$, we have $\tau_d = q_{2^{l-i-1}(2s+1)}$. Denoting the competitive ratio at time d by $apx(d)$, we prove that

$$apx(d) = \frac{\sigma_r}{(2^i + s + 1) \cdot \sum_{j=2^l+2^{l-i-1}(2s+1)}^{2^l-1+2^{l-i-1}(2s+2)} \frac{1}{j}}.$$

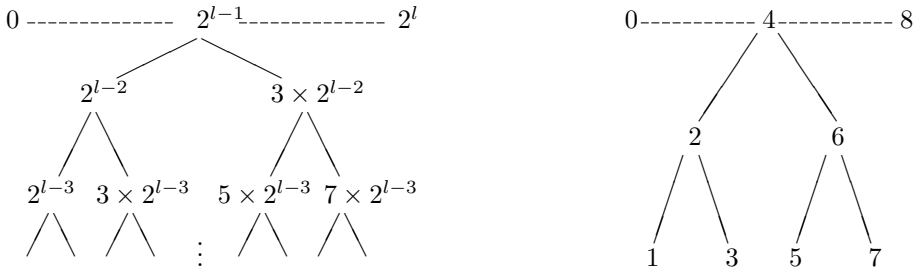


Figure 2 The left-hand side shows the top three levels of the binary tree for a general l, with $\tau = (q_{2^{l-1}}, q_{2^{l-2}}, q_{3\times 2^{l-2}}, q_{2^{l-3}}, q_{3\times 2^{l-3}}, q_{5\times 2^{l-3}}, q_{7\times 2^{l-3}}, \dots)$. The right-hand side shows the complete binary tree for $l = 3$, with $\tau = (q_4, q_2, q_6, q_1, q_3, q_5, q_7)$.

Writing $apx(d)$ as $apx(i, s)$, we prove that, fixing i, $apx(i, s)$ is strictly increasing in s; and letting $s = 2^i - 1$, $apx(i, s)$ is strictly increasing in i. Thus the worst competitive ratio occurs at $i = l-1$ and $s = 2^{l-1} - 1$. As $apx(l-1, 2^{l-1}-1) = (2 - \frac{1}{2^l})\sigma_r < 2\sigma_r$, Lemma 7 holds. ◀

The theorem below follows easily from the above two lemmas.

▶ **Theorem 8.** *There exists a deterministic polynomial-time online algorithm for the ATWC problem, whose competitive ratio can be arbitrarily close to* $2\ln 2$. *Moreover, the running time is polynomial in* $\frac{1}{\epsilon}$ *for competitive ratio* $2\ln 2 + \epsilon$.

Remark. When the number of points is large but the maximum number m of simultaneously present points is small, the running time of the algorithm for each arriving point is polynomial in m and can be much faster than being polynomial in the size of the input.

Following Theorem 5, Algorithm 1 is essentially optimal. Inspired by our constructions of Q and τ, we actually characterize the optimal solution for the online ATWC problem, whose competitive ratio is exactly $2\ln 2$: see Theorem 9. However, this solution involves irrational numbers and cannot be exactly computed in polynomial time.

▶ **Theorem 9.** *For any integer* $d = 2^i + s$ *with* $i \geq 0$ *and* $0 \leq s \leq 2^i - 1$, *let* $\tau_d = \log_2(1 + \frac{2s+1}{2^{i+1}})$. *If Algorithm 1 creates the d-th new position in* $\hat{Q}$ *to be* τ_d, *the competitive ratio is* $2\ln 2$.

4 The 2-Dimensional Online All-Time Worst-Case Problem

We now consider the 2-dimensional online ATWC problem in a square – without loss of generality, $P = [0, 1]^2$. One difficulty is that, different from the 1-dimensional problem where it is trivial to have $Disp(n; P) = \frac{1}{n+1}$ for any $n \geq 1$, here neither $Disp(n; P)$ nor $Pack(r; P)$ has a known closed-form optimal solution (whether polynomial-time computable or not). Accordingly, our lower-bound and our competitive algorithm must rely on some proper upper- and lower-bounds for $Disp(n; P)$, which is part of the reason why the resulting bounds are not tight. In particular, we have the following.

▶ **Lemma 10.** *For any* $n \geq 1$, $\frac{2}{5+\sqrt{2\sqrt{3}n}} \leq Disp(n; P) \leq \frac{2}{2+\sqrt{2\sqrt{3}n}}$.

4.1 The Lower Bound

Interestingly, not only the dispersion problem is closely related to *uniform* packing (i.e., the disks all have the same radius) as we have seen in Section 2, but we also obtain a lower bound for the online ATWC problem by carefully fitting a *non-uniform* packing into the square.

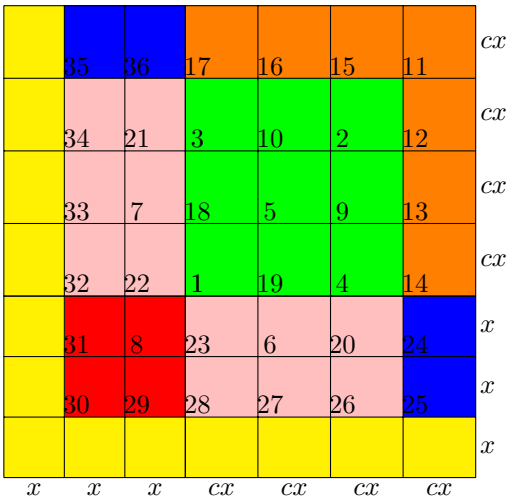


Figure 3 The set of pre-fixed positions, $Q = \{q_1, \ldots, q_{36}\}$, and the grid created by Q. A grid point labelled by i indicates the position q_i. The colored areas are used in the algorithm's description. More specifically, denoting a rectangle by the position in Q at its lower-left corner, the green area is $(3, 10, 2; 18, 5, 9; 1, 19, 4)$; the two pink areas are $(23, 6, 20; 28, 27, 26)$ and $(34, 21; 33, 7; 32, 22)$; the red area is $(31, 8; 30, 29)$; the two blue areas are $(35, 36)$ and $(24, 25)$; the orange area is $(17, 16, 15, 11, 12, 13, 14)$; and finally the yellow area contains all the remaining rectangles: that is, rectangles adjacent to the left boundary and the bottom boundary.

The idea is to imagine each position created in an online algorithm as a disk centered at that position. The radius of each disk is a function of the algorithm's competitive ratio and the optimal solutions to specific dispersion problems without time. Note that the area covered by the disks is upper-bounded by the area of the square containing them. Combining these relations together gives us the following theorem.

▶ **Theorem 11.** *No online algorithm achieves a competitive ratio better than 1.183 for the 2-dimensional ATWC problem in a square.*

4.2 A Polynomial-Time Online Algorithm

Now we provide a deterministic polynomial-time online algorithm which is 1.591-competitive to OPT_A. Similar to Algorithm 1, we construct a set Q of pre-fixed positions. However, it is unclear how to define Q of arbitrary size in the square, and we construct a set of 36 positions, denoted by $Q = \{q_1, \ldots, q_{36}\}$. It depends on a parameter $1 < c < \sqrt{2}$ and $x = \frac{1}{3+4c}$, as in Figure 3. The q_i's indices specify the order according to which they should be occupied, thus we do not need an extra ordering τ. Note these positions create a grid in P and split it into multiple rectangles. The choice of c (and x, Q) will become clear in the analysis.

Whenever a new position needs to be created, we pick the first position in Q that has never been occupied yet. When all positions in Q are occupied, we may (1) create a new position in the center of a current rectangle with the largest area, split this rectangle into four smaller ones accordingly, and add the vertices of the new rectangles into the grid; or (2) create a new position at a grid point that has never been occupied yet. The main Algorithm 2 is similar to Algorithm 1. It uses in Step 8 a sub-routine to implement (1) and (2) above: the *Position Creation Phase,* as defined in [8], where we further provide some intuition on the choices of Q, x, and c. Setting $c = 1.271$, we have the following.

Algorithm 2. A polynomial-time online algorithm for the ATWC problem in a square.

Parameter: c such that $1 < c < \sqrt{2}$, the corresponding $x = \frac{1}{3+4c}$, and Q.
Input: A sequence of points arriving and departing along time.

Denote by $\hat{Q}$ the set of positions ever occupied by a point. At any point of time, a position in $\hat{Q}$ is labeled *occupied* if currently there is a point there and *vacant* otherwise. Initially $\hat{Q} = \emptyset$.
When a point w leaves, change the label of its position in $\hat{Q}$ from *occupied* to *vacant*.
When a point w arrives:
if $\hat{Q} = \emptyset$ or all positions in $\hat{Q}$ are labelled *occupied* **then**
  if $|\hat{Q}| < 36$ **then**
    Put w at position $q_{|\hat{Q}|+1}$, add this position to $\hat{Q}$ and label it *occupied*.
  else
    Compute a position q according to the Position Creation Phase defined in [8].
    Put w at position q, add q to $\hat{Q}$ and label it *occupied*.
  end if
else
  Arbitrarily choose a vacant position q from $\hat{Q}$ and label it *occupied*.
  Put w at position q.
end if

▶ **Theorem 12.** *Algorithm 2 runs in polynomial time and is* 1.591*-competitive for the 2-dimensional online ATWC problem in a square.*

Note that the upper-bound for $Disp(n; P)$ in Lemma 10 is not tight when n is small. With better upper-bounds for $Disp(n; P)$, better competitive ratios for our algorithm can be directly obtained via a similar analysis. Moreover, we believe the competitive ratio can be improved by using a larger set Q and the best ordering for positions in Q. Such a Q and a rigorous analysis based on it are left for future studies. Finally, similar techniques can be used when P is a rectangle, but the gap between the lower- and upper-bounds will be even larger, and the analysis will be more complicated without adding much new insight to the problem. Thus we leave a thorough study on rectangles for the future.

5 The General k-Dimensional Online ATWC Problem

Although the literature gives us little understanding about the optimal dispersion/packing problem in an arbitrary k-dimensional polytope P with $k \geq 2$, we are still able to provide a simple lower-bound and a simple polynomial-time algorithm for the online ATWC problem. Below we only state the theorems.

▶ **Theorem 13.** *For any $k \geq 2$, no online algorithm achieves a competitive ratio better than $\frac{7}{6}$ for the ATWC problem for arbitrary polytopes.*

For any polytope P, letting the *covering rate* be the ratio between the edge-lengths of the maximum inscribed cube and the minimum bounding cube, we have the following theorem. Note that, although a natural greedy algorithm provides a 2-competitive ratio, the exact greedy solution may not be computable in polynomial time. Here we show the greedy algorithm can be efficiently approximated arbitrarily closely. The geometric problems of finding the minimum bounding cube, deciding whether a position is in P, and finding the distance between a point in P and the boundary of P are given as oracles.

Algorithm 3. Algorithm $\mathcal{A}_{\mathcal{I}}$ for computing $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2\}$ satisfying properties Φ.1 and Φ.2.

Input: A sequence $S = ((s_1, d_1), \ldots, (s_n, d_n))$.

Let $\mathcal{I}_1 = \mathcal{I}_2 = \emptyset$, $s = -1$, $d = 0$ and $T = \max_{i\in[n]} d_i$. (s and d are end-points of a "sliding window" for the arriving times under consideration.)
Let $index = 1$.
while d $\neq T$ **do**
  Let $\hat{S} = \{i | i \in S, s_i > s, s_i \leq d, d_i > d\}$.
  if $\hat{S} \neq \emptyset$ **then**
    Arbitrarily choose $j \in \arg\max_{i\in\hat{S}} d_i$ and add j to $\mathcal{I}_{index}$; $s = d$; then $d = d_j$.
  else
    $s = d$; then $d = \min_{i\in S, s_i > d} s_i$.
  end if
  $index = 3 - index$.
end while
Output $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2\}$.

▶ **Theorem 14.** *For any constants $\gamma, \epsilon > 0$, integer $k \geq 2$ and k-dimensional polytope P with covering rate at least γ, there exists a deterministic polynomial-time online algorithm for the ATWC problem, with competitive ratio $\frac{2}{1-\epsilon}$ and running time polynomial in $\frac{1}{(\gamma\epsilon)^k}$.*

6 The General k-Dimensional Offline CD Problem

By Claim 1, no online algorithm provides a good competitive ratio for the CD problem, thus we focus on the offline problem. Given an input sequence $S = ((s_1, d_1), \ldots, (s_n, d_n))$, we first slice the whole time interval $[0, T]$ into smaller ones by the arriving time s_i and the departure time d_i of each point $i \in [n]$. Thus the set of present points only changes at the end-points of the intervals and stays the same within an interval. Our algorithm will be such that, *in each time interval,* the minimum distance is a good approximation to the optimal dispersion problem without time, for the points present in this interval.

Interestingly, this is achieved by reducing the offline CD problem to the online ATWC problem, for any dimension k and polytope P. To carry out this idea, we first provide a polynomial-time algorithm $\mathcal{A}_{\mathcal{I}}$ (Algorithm 3) that, given a sequence S, selects a subset $\mathcal{I}$ of points from S. The set $\mathcal{I}$ satisfies the following properties, as proved in [8].

(Φ.1) $\mathcal{I}$ can be partitioned into two groups $\mathcal{I}_1$ and $\mathcal{I}_2$ such that the points in the same group have disjoint time intervals.

(Φ.2) For any time $0 \leq t \leq T$, if there are points in S present at time t, then at least one of them is selected to $\mathcal{I}$.

The offline CD algorithm $\mathcal{A}_{CD}$ uses algorithm $\mathcal{A}_{\mathcal{I}}$ to select $\mathcal{I}$ from its input S, eliminates the selected points from S, and repeats on the remaining S. Recall that m is the maximum number of points simultaneously present at any time. By property Φ.2, this procedure ends in at most m iterations. Based on the partitions constructed by $\mathcal{A}_{\mathcal{I}}$, $\mathcal{A}_{CD}$ constructs an instance of the online ATWC problem and uses any online algorithm $\mathcal{A}_{ATWC}$ for the latter as a black-box, so as to decide how to locate the points. Algorithm $\mathcal{A}_{CD}$ is defined in Algorithm 4 and we have the following theorem. Below we sketch the main ideas.

Algorithm 4. $\mathcal{A}_{CD}$

Input: A sequence $S = ((s_1, d_1), \ldots, (s_n, d_n))$.

1: Let $r = 0$.
2: **while** $S \neq \emptyset$ **do**
3: Run $\mathcal{A}_{\mathcal{I}}$ on S to obtain two disjoint sets $\mathcal{I}_{2r+1}, \mathcal{I}_{2r+2} \subseteq S$.
4: $S = S \setminus (\mathcal{I}_{2r+1} \cup \mathcal{I}_{2r+2})$.
5: $r = r + 1$.
6: **end while**
7: Run $\mathcal{A}_{ATWC}$ on the following online sequence of $2r$ points: for all $i \in \{0, 1, \ldots, r-1\}$, points $2i+1$ and $2i+2$ arrive at time i. All points depart at time r.
8: Letting x_{2i+1}, x_{2i+2} be the two positions returned by $\mathcal{A}_{ATWC}$ at time i, assign all points in $\mathcal{I}_{2i+1}$ to x_{2i+1} and all points in $\mathcal{I}_{2i+2}$ to x_{2i+2}.

▶ **Theorem 15.** *$\forall\ k \geq 1$ and k-dimensional polytope P, given any polynomial-time σ-competitive online algorithm $\mathcal{A}_{ATWC}$ for ATWC, there is a polynomial-time offline algorithm $\mathcal{A}_{CD}$ for CD with competitive ratio $\sigma \max_{i \geq 1} \frac{Disp(i;P)}{Disp(2i;P)} \leq 2\sigma$, using $\mathcal{A}_{ATWC}$ as a black-box.*

Proof Ideas. Given an input sequence S, we slice the whole time interval $[0, T]$ into smaller ones according to the arriving time and the departure time of each point. Denote these small intervals by $T_1, \ldots, T_l$, where l is the number of small intervals created. For each interval T_i, let S_i be the set of points that overlap with T_i and $n_i = |S_i|$. By properties $\Phi.1$ and $\Phi.2$, all points in S_i are eliminated from S in the first n_i iterations of $\mathcal{A}_{\mathcal{I}}$, thus are located at the first $2n_i$ positions created by $\mathcal{A}_{ATWC}$. The minimum distance among points in T_i (and to the boundary) is at least $\frac{Disp(2n_i;P)}{\sigma}$, since algorithm $\mathcal{A}_{ATWC}$ has competitive ratio σ. Thus, within each T_i, the competitive ratio to the optimal solution is upper-bounded by $\frac{\sigma Disp(n_i;P)}{Disp(2n_i;P)}$. Taking summation over all T_i's, the competitive ratio is upper-bounded by $\sigma \max_{i \geq 1} \frac{Disp(i;P)}{Disp(2i;P)}$. Finally, we prove $\max_{i \geq 1} \frac{Disp(i;P)}{Disp(2i;P)} \leq 2$, finishing the proof of Theorem 15. ◀

References

1 Shimon Abravaya and Michael Segal. Maximizing the number of obnoxious facilities to locate within a bounded region. *Computers & Operations Research*, 37(1):163–171, 2010.

2 Christoph Baur and Sándor P. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30(3):451–470, 2001.

3 Boaz Ben-Moshe, Matthew J Katz, and Michael Segal. Obnoxious facility location: Complete service with minimal harm. *International Journal of Computational Geometry & Applications*, 10(06):581–592, 2000.

4 Marc Benkert, Joachim Gudmundsson, Christian Knauer, Esther Moet, René van Oostrum, and Alexander Wolff. A polynomial-time approximation algorithm for a geometric dispersion problem. In *International Computing and Combinatorics Conference*, pages 166–175. Springer, 2006.

5 Benjamin Birnbaum and Kenneth J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica*, 55(1):42–59, 2009.

6 Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. Max-sum diversity via convex programming. In *Proceedings of the 32nd Symposium on Computational Geometry (SoCG)*, pages 26:1–26:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

7 Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. Local search for max-sum diversification. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 130–142. SIAM, 2017.

8 Jing Chen, Bo Li, and Yingkai Li. Efficient approximations for the online dispersion problem. *arXiv:1704.06823*, 2017. Full version.

9 Henry Cohn, Abhinav Kumar, Stephen D. Miller, Danylo Radchenko, and Maryna Viazovska. The sphere packing problem in dimension 24. *arXiv:1603.06518*, 2016.

10 Lester E. Dubins and Edwin H. Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68(1):1–17, 1961.

11 Adrian Dumitrescu and Minghui Jiang. Dispersion in disks. *Theory of Computing Systems*, 51(2):125–142, 2012.

12 Leah Epstein and Rob Van Stee. Optimal online algorithms for multidimensional packing problems. *SIAM Journal on Computing*, 35(2):431–448, 2005.

13 Leah Epstein and Rob Van Stee. Bounds for online bounded space hypercube packing. *Discrete optimization*, 4(2):185–197, 2007.

14 L. Fejes Tóth. Über die dichteste kugellagerung. *Mathematische Zeitschrift*, 48(1):676–684, 1942.

15 Sándor P. Fekete and Henk Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38(3):501–511, 2004.

16 Dimitris Fotakis. Incremental algorithms for facility location and k-median. *Theoretical Computer Science*, 361(2):275–313, 2006.

17 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

18 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12(3):133–137, 1981.

19 Eric Friedman, Christos-Alexandros Psomas, and Shai Vardi. Dynamic fair division with minimal disruptions. In *Proceedings of the sixteenth ACM conference on Economics and Computation*, pages 697–713. ACM, 2015.

20 Thomas C. Hales. A proof of the Kepler conjecture. *Annals of mathematics*, 162(3):1065–1185, 2005.

21 Mhand Hifi and Rym M'hallah. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research*, 2009:150624:1–150624:22, 2009.

22 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

23 Pedro Hokama, Flávio K. Miyazawa, and Rafael C. S. Schouery. A bounded space algorithm for online circle packing. *Information Processing Letters*, 116(5):337–342, 2016.

24 Wenqi Huang and Tao Ye. Greedy vacancy search algorithm for packing equal circles in a square. *Operations Research Letters*, 38(5):378–382, 2010.

25 Matthew J. Katz, Klara Kedem, and Michael Segal. Improved algorithms for placing undesirable facilities. *Computers & Operations Research*, 29(13):1859–1872, 2002.

26 Marco Locatelli and Ulrich Raber. Packing equal circles in a square: a deterministic global optimization approach. *Discrete Applied Mathematics*, 122(1):139–166, 2002.

27 Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM Journal on Computing*, 32(3):816–832, 2003.

28 Adam Meyerson. Online facility location. In *42rd Annual IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2001.

29 Flávio K Miyazawa, Lehilton L. C. Pedrosa, Rafael C. S. Schouery, Maxim Sviridenko, and Yoshiko Wakabayashi. Polynomial-time approximation schemes for circle packing problems. In *European Symposium on Algorithms*, pages 713–724. Springer, 2014.

30 Ronald Peikert, Diethelm Würtz, Michael Monagan, and Claas de Groot. Packing circles in a square: a review and new results. In *System Modelling and Optimization: Proceedings of the 15th IFIP Conference, Zurich, Switzerland, September 2–6, 1991*, pages 45–54. Springer, 1992.

31 Zhongping Qin, Yinfeng Xu, and Binhai Zhu. On some optimization problems in obnoxious facility location. In *International Computing and Combinatorics Conference*, pages 320–329. Springer, 2000.

32 Sekharipuram S. Ravi, Daniel J. Rosenkrantz, and Giri K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

33 Claude Ambrose Rogers. Existence theorems in the geometry of numbers. *Annals of Mathematics*, pages 994–1002, 1947.

34 Daniel J. Rosenkrantz, Giri K. Tayi, and S. S. Ravi. Obtaining online approximation algorithms for facility dispersion from offline algorithms. *Networks*, 47(4):206–217, 2006.

35 J. Schaer. The densest packing of nine circles in a square. *Canad. Math. Bull*, 8:273–277, 1965.

36 J. Schaer and A. Meir. On a geometric extremum problem. *Canad. Math. Bull*, 8:21–27, 1965.

37 B. L. Schwartz. Separating points in a square. *J. Recr. Math*, 3:195–204, 1970.

38 Steven S. Seiden. On the online bin packing problem. *Journal of the ACM (JACM)*, 49(5):640–671, 2002.

39 Hugo Steinhaus. The problem of fair division. *Econometrica*, 16:101–104, 1948.

40 Walter Stromquist. How to cut a cake fairly. *The American Mathematical Monthly*, 87(8):640–644, 1980.

41 Péter Gábor Szabó and Eckard Specht. Packing up to 200 equal circles in a square. In *Models and Algorithms for Global Optimization*, pages 141–156. Springer, 2007.

42 Maryna Viazovska. The sphere packing problem in dimension 8. *arXiv:1603.04246*, 2016.

Online Covering with Sum of ℓ_q-Norm Objectives*

Viswanath Nagarajan[1] and Xiangkun Shen[2]

1 **University of Michigan, Ann Arbor, MI, USA**
viswa@umich.edu
2 **University of Michigan, Ann Arbor, MI, USA**
xkshen@umich.edu

Abstract

We consider fractional online covering problems with ℓ_q-norm objectives. The problem of interest is of the form $\min\{f(x) : Ax \geq 1, x \geq 0\}$ where $f(x) = \sum_e c_e \|x(S_e)\|_{q_e}$ is the weighted sum of ℓ_q-norms and A is a non-negative matrix. The rows of A (i.e. covering constraints) arrive online over time. We provide an online $O(\log d + \log \rho)$-competitive algorithm where $\rho = \frac{\max a_{ij}}{\min a_{ij}}$ and d is the maximum of the row sparsity of A and $\max |S_e|$. This is based on the online primal-dual framework where we use the dual of the above convex program. Our result expands the class of convex objectives that admit good online algorithms: prior results required a monotonicity condition on the objective f which is not satisfied here. This result is nearly tight even for the linear special case. As direct applications we obtain (i) improved online algorithms for non-uniform buy-at-bulk network design and (ii) the first online algorithm for throughput maximization under ℓ_p-norm edge capacities.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases online algorithm, covering/packing problem, convex, buy-at-bulk, throughput maximization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.12

1 Introduction

The online primal-dual approach is a widely used approach for online problems. This involves solving a discrete optimization problem online as follows (i) formulate a linear programming relaxation and obtain a primal-dual online algorithm for it; (ii) obtain an online rounding algorithm for the resulting fractional solution. While this is similar to a linear programming (LP) based approach for offline optimization problems, a key difference is that solving the LP relaxation in the online setting is highly non-trivial. (Recall that there are general polynomial time algorithms for solving LPs offline.) So there has been a lot of effort in obtaining good online algorithms for various classes of LPs: see [1, 13, 23] for pure covering LPs, [13] for pure packing LPs and [5] for certain mixed packing/covering LPs. Such online LP solvers have been useful in obtaining online algorithms for various problems, eg. set cover [2], facility location [1], machine scheduling [5], caching [8] and buy-at-bulk network design [21].

Recently, [6] initiated a systematic study of online fractional covering and packing with *convex* objectives; see also the full versions [7, 11, 15]. These papers obtained good online algorithms for a large class of fractional convex covering problems. They also demonstrated the utility of this approach via many applications that could not be solved using just online LPs. However these results were limited to convex objectives $f : \mathbb{R}^n_+ \to \mathbb{R}_+$ satisfying a

* A full version of the paper is available at https://arxiv.org/abs/1705.02194.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 12; pp. 12:1–12:12

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

monotone gradient property, i.e. $\nabla f(z) \geq \nabla f(y)$ pointwise for all $z, y \in \mathbb{R}^n$ with $z \geq y$. There are however many natural convex functions that do not satisfy such a gradient monotonicity condition. Note that this condition requires the Hessian $\nabla^2 f(x)$ to be pointwise non-negative in addition to convexity which only requires $\nabla^2 f(x)$ to be positive semidefinite.

In this paper, we focus on convex functions f that are sums of different ℓ_q-norms. This is a canonical class of convex functions with non-monotone gradients and prior results are not applicable; see Section 1.1 for a more detailed comparison. We show that sum of ℓ_q-norm functions admit a logarithmic competitive online algorithm. This result is nearly tight because there is a logarithmic lower bound even for online covering LPs (which corresponds to an ℓ_1 norm objective). We also provide two applications of our result (i) improved competitive ratios (by two logarithmic factors) for some online non-uniform buy-at-bulk problems studied in [21], and (ii) the first online algorithm for throughput maximization with ℓ_p-norm edge capacities (the competitive ratio is logarithmic which is known to be best possible even in the special case of individual edge capacities).

Given that we achieve log-competitive online algorithms for sums of ℓ_q-norms, a natural question is whether such a result holds for all norms. Recall that any norm is a convex function. It turns out that a log-competitive algorithm is not possible for general norms. This follows from a result in [7] which shows an $\Omega(q \log d)$ lower bound for minimizing the objective $\|Bx\|_q$ under covering constraints (where B is a non-negative matrix). It is still an interesting open question to identify the correct competitive ratio for general norm functions.

1.1 Our Results and Techniques

We consider the online covering problem

$$\min \left\{ \sum_{e=1}^{r} c_e \|x(S_e)\|_{q_e} \,:\, Ax \geq \mathbf{1},\, x \in \mathbb{R}^n_+ \right\}, \tag{1}$$

where each $S_e \subseteq [n] := \{1, 2, \cdots n\}$, $q_e \geq 1$, $c_e \geq 0$ and A is a non-negative $m \times n$ matrix. For any $S \subseteq [n]$ and $q \geq 1$ we use the standard notation $\|x(S)\|_q = \left(\sum_{i \in S} x_i^q\right)^{1/q}$. We also consider the dual of this convex program, which is the following packing problem:

$$\max \left\{ \sum_{k=1}^{m} y_k \,:\, A^T y = \mu,\, \sum_{e=1}^{r} \mu_e = \mu,\, \|\mu_e(S_e)\|_{p_e} \leq c_e \,\forall e \in [r],\, y \geq 0 \right\}. \tag{2}$$

The values p_e above satisfy $\frac{1}{p_e} + \frac{1}{q_e} = 1$; so $\|\cdot\|_{p_e}$ is the dual norm of $\|\cdot\|_{q_e}$. This dual can be derived from (1) using Lagrangian duality.

Our framework captures the classic setting of packing/covering LPs when $r = n$ and for each $e \in [n]$ we have $S_e = \{e\}$ and $q_e = 1$. Our main result is:

▶ **Theorem 1.** *There is an $O(\log d + \log \rho)$-competitive online algorithm for (1) and (2) where the covering constraints in (1) and variables y in (2) arrive over time. Here d is the maximum of the row-sparsity of A and $\max_{e=1}^{r} |S_e|$ and $\rho = \frac{\max\{a_{ij}\}}{\min\{a_{ij}\}}$.*

We note that this bound is also the best possible, even in the linear case [13]. For just the covering problem, a better $O(\log d)$ bound is known in the linear case [23] as well as for convex functions with monotone gradients [6].

The algorithm in Theorem 1 is the natural extension of the primal-dual approach for online LPs [13]. We use the gradient $\nabla f(x)$ at the current primal solution x as the cost function, and use this to define a multiplicative update for the primal. Simultaneously, the

dual solution y is increased additively. This algorithm is in fact identical to the one in [6] for convex functions with monotone gradients. See Algorithm 1 for the formal description. The contribution of this paper is in the analysis of this algorithm, which requires new ideas to deal with non-monotone gradients.

Limitations of previous approaches [6]

Recall that the general convex covering problem is

$$\min \left\{ f(x) \,:\, Ax \geq \mathbf{1},\, x \in \mathbb{R}^n_+ \right\},$$

where $f : \mathbb{R}^n_+ \to \mathbb{R}_+$ is a convex function. Its dual is:

$$\max \left\{ \sum_{k=1}^{m} y_k - f^*(\mu) \,:\, A^T y = \mu,\, y \geq 0 \right\},$$

where $f^*(\mu) = \max_{x \in \mathbb{R}^n_+} \{\mu^T x - f(x)\}$ is the Fenchel conjugate of f. When f is the sum of ℓ_q-norms, these primal-dual convex programs reduce to (1) and (2).

We restrict the discussion of prior techniques to functions f with $\max_{x \in \mathbb{R}^n_+} \frac{x^T \nabla f(x)}{f(x)} \leq 1$ because this condition is satisfied by sums of ℓ_q norms.[1] At a high level, the analysis in [6] uses the gradient monotonicity to prove a *pointwise upper bound* $A^T y \leq \nabla f(\bar{x})$ where $\bar{x}$ is the final primal solution. This allows them to lower bound the dual objective by $\sum_{k=1}^{m} y_k$ because $f^*(\nabla f(\bar{x})) \leq 0$ for any $\bar{x}$ (see Lemma 4(d) in [6]). Moreover, proving the pointwise upper bound $A^T y \leq \nabla f(\bar{x})$ is similar to the task of showing dual feasibility in the *linear* case [13, 23] where $\nabla f(\bar{x})$ corresponds to the (fixed) primal cost coefficients.

Below we give a simple example with an ℓ_q-norm objective where the pointwise upper bound $A^T y \leq \nabla f(\bar{x})$ is not satisfied by the online primal-dual algorithm unless the dual solution y is scaled down by a large (i.e. polynomial) factor. This means that one cannot obtain a sub-polynomial competitive ratio for (1) using this approach directly.

Consider an instance with objective function $f(x) = \|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$. There are $m = \sqrt{n}$ covering constraints, where the k^{th} constraint is $\sum_{i=k(m-1)+1}^{km} x_i \geq 1$. Note that each variable appears in only one constraint. Let P be the value of primal objective and D be the value of dual objective at any time. Suppose that the rate of increase of the primal objective is at most α times that of the dual; α corresponds to the competitive ratio in the online primal-dual algorithm. Upon arrival of any constraint k, it follows from the primal updates that all the variables $\{x_i\}_{i=k(m-1)+1}^{km}$ increase from 0 to $\frac{1}{m}$. So the increase in P due to constraint k is $(\sqrt{k} - \sqrt{k-1})\frac{1}{\sqrt{m}}$ for iteration k. This means that the increase in D is at least $\frac{1}{\alpha}(\sqrt{k} - \sqrt{k-1})\frac{1}{\sqrt{m}}$, and so $y_k \geq \frac{1}{\alpha}(\sqrt{k} - \sqrt{k-1})\frac{1}{\sqrt{m}}$. Finally, since $\bar{x} = \frac{1}{m}\mathbf{1}$, we know that $\nabla f(\bar{x}) = \frac{1}{m}\mathbf{1}$ (recall $n = m^2$). On the other hand, $(A^T y)_1 = y_1 \geq \frac{1}{\alpha\sqrt{m}}$. Therefore, in order to guarantee $A^T y \leq \nabla f(\bar{x})$ we must have $\alpha \geq \sqrt{m} = n^{1/4}$.

Our approach

First, we show that by duplicating variables and using an online separation oracle approach (as in [1]) one can ensure that the sets $\{S_e\}_{e=1}^{r}$ are disjoint. This allows for a simple

[1] The result in [6] also applies to other convex functions with monotone gradients, but the competitive ratio depends exponentially on $\max_{x \in \mathbb{R}^n_+} \frac{x^T \nabla f(x)}{f(x)}$.

expression for ∇f which is useful in the later analysis. Then we utilize the specific form of the primal-dual convex programs (1) and (2) and an explicit expression for ∇f to show that the dual y is approximately feasible. In particular we show that $\|y^T A(S_e)\|_{p_e} \leq O(\log d\rho) \cdot c_e$ for each $e \in [r]$; here $A(S_e)$ denotes the submatrix of A with columns from S_e. Note that this is a weaker requirement than upper bounding $A^T y$ pointwise by $\nabla f(\bar{x})$.

In order to bound $\|y^T A(S_e)\|_{p_e}$, we analyze each $e \in [r]$ separately. We partition the steps of the algorithm into *phases* where phase j corresponds to steps where $\Phi_e = \sum_{i \in S_e} x_i^{q_e} \approx \theta^j$; here $\theta > 1$ is a parameter that depends on q_e. The number of phases can be bounded using the fact that Φ_e is monotonically increasing. By triangle inequality we upper bound $\|y^T A(S_e)\|_{p_e}$ by $\sum_j \|y_{(j)}^T A(S_e)\|_{p_e}$ where $y_{(j)}$ denotes the dual variables that arrive in phase j. And in each phase j, we can upper bound $\|y_{(j)}^T A(S_e)\|_{p_e}$ using the differential equations for the primal and dual updates.

Applications

We also provide two applications of Theorem 1.

Non-uniform multicommodity buy-at-bulk. This is a well-studied network design problem in the offline setting [16, 17]. For its online version, the first poly-logarithmic competitive ratio was obtained recently in [21]. A key step in this result was a fractional online algorithm for a certain mixed packing-covering LP. We improve the competitive ratio of this step from $O(\log^3 n)$ to $O(\log n)$ which leads to a corresponding improvement in the final result of [21]. See Theorem 6.

Throughput maximization with ℓ_p-norm capacities. The online problem of maximizing throughput subject to edge capacities is well studied and a tight logarithmic competitive ratio is known [4, 13]. We consider the generalization where instead of individual edge capacities, we can have capacity constraints on subsets as follows. A ℓ_p-norm capacity of c for some subset S of edges means that the ℓ_p-norm of the loads on edges of S must be at most c. We show that one can obtain a randomized log-competitive algorithm even in this setting, which generalizes the case with edge-capacities. See Theorem 7.

1.2 Related Work

The online primal-dual framework for linear programs [14] is fairly well understood. Tight results are known for the class of packing and covering LPs [13, 23], with competitive ratio $O(\log d)$ for covering LPs and $O(\log d\rho)$ for packing LPs; here d is the row-sparsity and ρ is the ratio of the maximum to minimum entries in the constraint matrix. Such LPs are very useful because they correspond to the LP relaxations of many combinatorial optimization problems. Combining the online LP solver with suitable online rounding schemes, good online algorithms have been obtained for many problems, eg. set cover [2], group Steiner tree [1], caching [8] and ad-auctions [12]. Online algorithms for LPs with mixed packing and covering constraints were obtained in [5]; the competitive ratio was improved in [6]. Such mixed packing/covering LPs were also used to obtain an online algorithm for capacitated facility location [5]. A more complex mixed packing/covering LP was used recently in [21] to obtain online algorithms for non-uniform buy-at-bulk network design: as an application of our result, we obtain a simpler and better (by two log-factors) online algorithm for this problem.

There have also been a number of results utilizing the online primal-dual framework with *convex* objectives for specific problems, eg. matching [19], caching [25], energy-efficient scheduling [18, 22] and welfare maximization [10, 24]. All of these results involve separable convex/concave functions. Recently, [6] considered packing/covering problems with general (non-separable) convex objectives, but (as discussed previously) this result requires a monotone gradient assumption on the convex function. The sum of ℓ_q-norm objectives considered in this paper does not satisfy this condition. While our primal-dual algorithm is identical to [6], we need new techniques in the analysis.

All the results above (as well as ours) involve convex objectives and linear constraints. We note that [20] obtained online primal-dual algorithms for certain semidefinite programs (i.e. involving non-linear constraints). While both our result and [20] generalize packing/covering LPs, they are not directly comparable.

We also note that online algorithms with ℓ_q-norm objectives have been studied previously for many scheduling problems, eg. [3, 9]. These results use different approaches and are not directly comparable to ours. More recently [7] used ideas from the online primal-dual approach in an online algorithm for unrelated machine scheduling with ℓ_p-norm objectives as well as startup costs. However, the algorithm in [7] was tailored to their scheduling setting and we do not currently see a connection between our result and [7].

2 Preliminaries

Recall the primal covering problem (1) and its dual packing problem (2). In the online setting, the constraints in the primal and variables in the dual arrive over time. We need to maintain monotonically increasing primal (x) and dual (y) solutions. The following lemma shows that one can assume that the sets $\{S_e\}_{e=1}^r$ are *disjoint* without loss of generality. We defer the proof of the lemma to the full version. This leads to a much simpler expression for ∇f that will be used in Section 3.

▶ **Lemma 2.** *If there is a poly-time α-competitive algorithm for instances with disjoint S_e, then there is a poly-time $O(\alpha)$-competitive algorithm for general instances.*

Henceforth we will assume that the sets $\{S_e\}_{e=1}^r$ are disjoint. The dual program (2) in this case reduces to:

$$\max \left\{ \sum_{k=1}^{m} y_k \;:\; A^T y = \mu,\ \|\mu(S_e)\|_{p_e} \le c_e \ \forall e \in [r],\ y \ge 0 \right\}. \tag{3}$$

It is easy to see that weak duality holds (Lemma 3). Strong duality also holds because (1) satisfies Slater's condition; however we do not use this fact.

▶ **Lemma 3.** *For any pair of feasible solutions x to (1) and (y, μ) to (3), we have*

$$\sum_{e=1}^{r} c_e \|x(S_e)\|_{q_e} \ge \sum_{k=1}^{m} y_k.$$

Proof. This follows from the following inequalities:

$$\sum_{k=1}^{m} y_k = y^T \mathbf{1} \le y^T A x = \mu^T x \le \sum_{e=1}^{r} \sum_{i \in S_e} \mu_i \cdot x_i \le \sum_{e=1}^{r} \|\mu(S_e)\|_{p_e} \cdot \|x(S_e)\|_{q_e} \le \sum_{e=1}^{r} c_e \cdot \|x(S_e)\|_{q_e}.$$

The first inequality is by primal feasibility; the second and last are by dual feasibility; the fourth is by Hölder's inequality. ◀

When the k^{th} request $\sum_{i=1}^{n} a_{ki}x_i \geq 1$ arrives
Let τ be a continuous variable denoting the current time.;
while *the constraint is unsatisfied, i.e.,* $\sum_{i=1}^{n} a_{ki}x_i < 1$ **do**
  For each i with $a_{ki} > 0$, increase x_i at rate $\frac{\partial x_i}{\partial \tau} = \frac{a_{ki}x_i + \frac{1}{d}}{\nabla_i f(x)} = \frac{a_{ki}x_i + \frac{1}{d}}{c_e x_i^{q_e - 1}} \|x(S_e)\|_{q_e}^{q_e - 1}$;
  Increase y_k at rate $\frac{\partial y_k}{\partial \tau} = 1$;
  Set $\mu = A^T y$;
end

Algorithm 1: Algorithm for ℓ_q-norm packing/covering.

3 Algorithm and analysis

Let $f(x) = \sum_{e=1}^{r} c_e \|x(S_e)\|_{q_e}$ denote the primal objective in (1).

In order to ensure that the gradient ∇f is positive, the primal solution x starts off as $\delta \cdot \mathbf{1}$ where $\delta > 0$ is arbitrarily small. So we assume that the initial primal value is zero.

It is clear that the algorithm maintains a feasible and monotonically non-decreasing primal solution x. The dual solution (y, μ) is also monotonically non-decreasing, but not necessarily feasible. We will show that (y, μ) is $O(\log \rho d)$-approximately feasible, i.e. the packing constraints in (3) are violated by at most an $O(\log \rho d)$ factor.

▶ **Lemma 4.** *The primal objective $f(x)$ is at most twice the dual objective $\sum_{k=1}^{m} y_k$.*

Proof. We will show that the rate of increase of the primal is at most twice that of the dual. Consider the algorithm upon the arrival of some constraint k. Then

$$\frac{df(x)}{d\tau} = \sum_{i:a_{ki}>0} \nabla_i f(x) \cdot \frac{\partial x_i}{\partial \tau} = \sum_{i:a_{ki}>0} (a_{ki}x_i + \frac{1}{d}) \leq 2.$$

The inequality comes from the fact that (i) the process for the kth constraint is terminated when $\sum_i a_{ki}x_i = 1$ and (ii) the number of non-zeroes in constraint k is at most d. Also it is clear that the dual objective increases at rate one, which finishes the proof. ◀

▶ **Lemma 5.** *The dual solution (y, μ) is $O(\log \rho d)$-approximately feasible, i.e.*

$$\|\mu(S_e)\|_{p_e} \;\leq\; O(\log \rho d) \cdot c_e, \qquad \forall e \in [r].$$

Proof. Fix any $e \in [r]$. When $q_e = 1$ the analysis is simple (details are deferred to the full version). Here we only consider $q_e > 1$ which is the main part of the analysis. In order to prove the desired upper bound on $\|\mu(S_e)\|_{p_e}$ we use a potential function $\Phi = \sum_{i \in S_e} (x_i^{q_e})$. Let phase zero denote the period where $\Phi \leq \zeta := (\frac{1}{\max\{a_{ij}\} \cdot d^2})^{q_e}$, and for each $j \geq 1$, phase j is the period where $\theta^{j-1} \cdot \zeta \leq \Phi < \theta^j \cdot \zeta$. Here $\theta > 1$ is a parameter depending on q_e that will be determined later. Note that $\Phi \leq d(\frac{1}{\min\{a_{ij}\}})^{q_e}$ as variable x_i will never be increased beyond $1/\min_{j=1}^{m} a_{ij}$. So the number of phases is at most $3q_e \cdot \log(d\rho)/\log\theta$. Next, we bound the increase in $\|\mu(S_e)\|_{p_e}$ for each phase separately.

For any phase, we have the following equalities

$$\frac{\partial x_i}{\partial \tau} = \frac{a_{ki}x_i + \frac{1}{d}}{c_e x_i^{q_e - 1}} \|x(S_e)\|_{q_e}^{q_e - 1}, \quad \frac{\partial y_k}{\partial \tau} = 1, \quad \frac{\partial \mu_i}{\partial \tau} = a_{ki}$$

$$\Rightarrow d\mu_i = \frac{c_e\, a_{ki}\, x_i^{q_e - 1}}{(\sum_{j \in S_e} x_j^{q_e})^{1 - \frac{1}{q_e}} (a_{ki}x_i + \frac{1}{d})} dx_i \tag{4}$$

Phase zero. Suppose that each x_i increases to α_i in phase zero. From (4) we have

$$d\mu_i \leq \frac{d\, c_e\, a_{ki}\, x_i^{q_e-1}}{(\sum_{j\in S_e} x_j^{q_e})^{1-\frac{1}{q_e}}} dx_i \qquad \Rightarrow \qquad \frac{1}{d\, c_e\, a_{ki}} d\mu_i \leq \frac{x_i^{q_e-1}}{(\sum_{j\in S_e} x_j^{q_e})^{1-\frac{1}{q_e}}} dx_i.$$

This means that the increase $\Delta\mu_i$ in μ_i can be bounded as:

$$\frac{1}{d\, c_e\, a_{ki}} \Delta\mu_i \leq \int_{\delta}^{\alpha_i} \frac{x_i^{q_e-1}}{(\sum_{j\in S_e} x_j^{q_e})^{1-\frac{1}{q_e}}} dx_i \leq \int_0^{\alpha_i} 1 dx_i \leq \alpha_i.$$

Since in phase zero, $\Phi \leq (\frac{1}{\max\{a_{ij}\}\cdot d^2})^{q_e}$, we know that each $\alpha_i \leq \frac{1}{\max\{a_{ij}\}\cdot d^2}$. So $\Delta\mu_i \leq \frac{c_e}{d}$ and at the end of phase zero, we have $\|\mu(S_e)\|_{p_e} \leq \|\mu(S_e)\|_1 \leq c_e$. The last inequality is because $d \geq \max_e |S_e|$.

Phase $j \geq 1$. Let Φ_0 and Φ_1 be the value of Φ at the beginning and end of this phase respectively. In phase j, suppose that each x_i increases from s_i to t_i. Then,

$$d\mu_i = \frac{c_e\, a_{ki}\, x_i^{q_e-1}}{(\sum_{j\in S_e} x_j^{q_e})^{1-\frac{1}{q_e}} (a_{ki} x_i + \frac{1}{d})} dx_i \leq \frac{c_e x_i^{q_e-2}}{(\sum_{j\in S_e} x_j^{q_e})^{1-\frac{1}{q_e}}} dx_i\,.$$

So the increase $\Delta\mu_i$ in μ_i during this phase is:

$$\Delta\mu_i \leq \int_{s_i}^{t_i} \frac{c_e x_i^{q_e-2}}{(\sum_{j\in S_e} x_j^{q_e})^{1-\frac{1}{q_e}}} dx_i.$$

Note that variables $x_{i'}$ for $i' \neq i$ can also increase in this phase: so we cannot directly bound the above integral. This is precisely where the potential Φ is useful. We know that throughout this phase, $\sum_{i\in S_e} x_i^{q_e} \geq \Phi_0$. So,

$$\Delta\mu_i \leq c_e \int_{s_i}^{t_i} \frac{x_i^{q_e-2}}{\Phi_0^{1-\frac{1}{q_e}}} dx_i = c_e \frac{t_i^{q_e-1} - s_i^{q_e-1}}{(q_e-1)\Phi_0^{1-\frac{1}{q_e}}} = c_e \frac{t_i^{q_e-1} - s_i^{q_e-1}}{(q_e-1)\Phi_0^{\frac{1}{p_e}}}.$$

Above we used the assumption that $q_e > 1$ in evaluating the integral. Now,

$$\begin{aligned}(\Delta\mu_i)^{p_e} &\leq \frac{c_e^{p_e}}{(q_e-1)^{p_e}\,\Phi_0} \cdot \left(t_i^{q_e-1} - s_i^{q_e-1}\right)^{p_e} \leq \frac{c_e^{p_e}}{(q_e-1)^{p_e}\,\Phi_0} \cdot \left(t_i^{(q_e-1)p_e} - s_i^{(q_e-1)p_e}\right) \\ &= \frac{c_e^{p_e}}{(q_e-1)^{p_e}\,\Phi_0} \cdot (t_i^{q_e} - s_i^{q_e})\,.\end{aligned}$$

The first inequality above uses the fact that $(z_1+z_2)^{p_e} \geq z_1^{p_e} + z_2^{p_e}$ for any $p_e \geq 1$ and $z_1, z_2 \geq 0$, with $z_1 = s_i^{q_e-1}$ and $z_2 = t_i^{q_e-1} - s_i^{q_e-1}$. The last equality uses $\frac{1}{p_e} + \frac{1}{q_e} = 1$.

We can now bound

$$\sum_{i\in S_e} (\Delta\mu_i)^{p_e} \leq \frac{c_e^{p_e}}{(q_e-1)^{p_e}\,\Phi_0} \cdot \sum_{i\in S_e} (t_i^{q_e} - s_i^{q_e}) = \frac{c_e^{p_e}}{(q_e-1)^{p_e}\,\Phi_0} (\Phi_1 - \Phi_0) \leq \frac{c_e^{p_e}}{(q_e-1)^{p_e}} (\theta - 1)\,.$$

Let $\mu_j \in \mathbb{R}^{|S_e|}$ denote the increase in $\mu(S_e)$ during phase j. It follows from the above that $\|\mu_j\|_{p_e} \leq \frac{c_e}{q_e-1}(\theta-1)^{1/p_e}$.

Combining across phases. Note that $\mu = \sum_{j\geq 0} \mu_j$. By triangle inequality, we have

$$\|\mu\|_{p_e} \leq \sum_{j\geq 0} \|\mu_j\|_{p_e} \leq c_e + \sum_{j\geq 1} \|\mu_j\|_{p_e} \leq c_e \left(1 + \frac{3q_e(\theta-1)^{1/p_e}}{(q_e-1)\log\theta} \cdot \log(d\rho)\right). \tag{5}$$

To complete the proof we show next that for any $q_e > 1$, there is some choice of $\theta > 1$ such that the right-hand-side above is $O(\log(d\rho)) \cdot c_e$.

- If $q_e \geq 2$ then setting $\theta = 2$, we have $\frac{3q_e}{(q_e-1)}(\theta-1)^{1/p_e}/\log\theta \leq 6$.
- If $1 < q_e < 2$ then set $\theta = 1 + (q_e-1)^{-\epsilon p_e}$, where $\epsilon = \frac{1}{-\log(q_e-1)} > 0$. We have

$$\frac{(\theta-1)^{\frac{1}{p_e}}}{\log\theta} \leq \frac{(\theta-1)^{\frac{1}{p_e}}}{\log(q_e-1)^{-\epsilon p_e}} = \frac{(q_e-1)^{-\epsilon}}{\log(q_e-1)^{-\epsilon p_e}} = \frac{(q_e-1)^{-\epsilon}}{-\epsilon p_e \log(q_e-1)} = \frac{(q_e-1)^{-\epsilon}}{p_e} = \frac{2}{p_e}.$$

The first inequality above uses that $\theta - 1 = (q_e-1)^{-\epsilon p_e} > 1$. Thus we have

$$\frac{3q_e(\theta-1)^{1/p_e}}{(q_e-1)\log\theta} \leq \frac{6q_e}{(q_e-1)p_e} = 6,$$

where the last equality uses $\frac{1}{p_e} + \frac{1}{q_e} = 1$.

So in either case we have that the right-hand-side of (5) is at most $(1+6\log(d\rho)) \cdot c_e$. ◀

Combining Lemmas 3, 4 and 5, we obtain Theorem 1.

4 Applications

4.1 Online Buy-at-Bulk Network Design

In the non-uniform buy-at-bulk problem, we are given a directed graph $G = (V, E)$ with a monotone subadditive cost function $g_e : \mathbb{R}_+ \to \mathbb{R}_+$ on each edge $e \in E$ and a collection $\{(s_i, t_i)\}_{i=1}^m$ of m source/destination pairs. The goal is to find an $s_i - t_i$ path P_i for each $i \in [m]$ such that the objective $\sum_{e\in E} g_e(load_e)$ is minimized; here $load_e$ is the number of paths using e. An equivalent view of this problem involves two costs c_e and ℓ_e for each edge $e \in E$ and the objective $\sum_{e\in\cup P_i} c_e + \sum_{e\in E} \ell_e \cdot load_e$. In the online setting, the pairs (s_i, t_i) arrive over time and we need to decide on the path P_i immediately after the i^{th} pair arrives. Recently, [21] gave a modular online algorithm for non-uniform buy-at-bulk with competitive ratio $O(\alpha\beta\gamma \cdot \log^5 n)$ where:

- α is the "junction tree" approximation ratio,
- β is the integrality gap of the natural LP for single-sink instances,
- γ is the competitive ratio of an online algorithm for single-sink instances.

See [21] for more details. One of the main components in this result was an $O(\log^3 n)$-competitive fractional online algorithm for a certain mixed packing/covering LP. Here we show that Theorem 1 can be used to provide a better (and tight) $O(\log n)$-competitive ratio. This leads to the following improvement:

▶ **Theorem 6.** *There is an $O(\alpha\beta\gamma \cdot \log^3 n)$-competitive ratio for non-uniform buy-at-bulk, where α, β, γ are as above.*

The LP relaxation. Let $\mathcal{T} = \{s_i, t_i : i \in [m]\}$ denote the set of all sources/destinations. For each $i \in [m]$ and root $r \in V$ variable z_{ir} denotes the extent to which both s_i and t_i route to/from r. For each $r \in V$ and $e \in E$, variable x_{er} denotes the extent to which edge e is used in the routing to root r. For each $r \in V$ and $u \in \mathcal{T}$, variables $\{f_{r,u,e} : e \in E\}$ represent a flow between r and u. [21] relied on solving the following LP:

$$\begin{aligned}
\min \quad & \sum_{r\in V}\sum_{e\in E} c_e \cdot x_{e,r} \;+\; \sum_{r\in V}\sum_{e\in E} \ell_e \cdot \sum_{u\in\mathcal{T}} f_{r,u,e} \\
\text{s.t.} \quad & \sum_{r\in V} z_{ir} \geq 1, \qquad \forall i \in [m] \\
& \{f_{r,s_i,e} : e \in E\} \text{ is a flow from } s_i \text{ to } r \text{ of } z_{ir} \text{ units}, \qquad \forall r \in V,\, i \in [m] \\
& \{f_{r,t_i,e} : e \in E\} \text{ is a flow from } r \text{ to } t_i \text{ of } z_{ir} \text{ units}, \qquad \forall r \in V,\, i \in [m] \\
& f_{r,u,e} \leq x_{e,r}, \qquad \forall u \in \mathcal{T},\, e \in E \\
& x, f, z \geq 0
\end{aligned}$$

The online algorithm in [21] for this LP has competitive ratio $O(D \cdot \log n)$ w.r.t. the optimal integral solution; here D is an upper bound on the length of any $s_i - t_i$ path (note that D can be as large as n). Using a height reduction operation, they could ensure that $D = O(\log n)$ while incurring an additional $O(\log n)$-factor loss in the objective. This lead to the $O(\log^3 n)$ factor for the fractional online algorithm. Here we provide an improved $O(\log n)$-competitive algorithm for this LP which does not require any bound on the path-lengths.

For any $r \in V$ and $u \in \mathcal{T}$, let $\mathsf{MC}(r, u)$ denote the $u - r$ (resp. $r - u$) minimum cut in the graph with edge capacities $\{f_{r,u,e} : e \in E\}$ if u is a source (resp. destination). By the max-flow min-cut theorem, it follows that $z_{ir} \leq \min\{\mathsf{MC}(r, s_i), \mathsf{MC}(r, t_i)\}$. Using this, we can combine the first three constraints of the above LP into the following:

$$\sum_{r\in V} \min\{\mathsf{MC}(r, s_i), \mathsf{MC}(r, t_i)\} \;\geq\; 1, \qquad \forall i \in [m].$$

For a fixed $i \in [m]$, this constraint is equivalent to the following. For each $r \in V$, pick either an $s_i - r$ cut (under capacities $f_{r,s_i,\star}$) or an $r - t_i$ cut (under capacities $f_{r,t_i,\star}$), and check if the total cost of these cuts is at least 1. This leads to the following reformulation that eliminates the x and z variables.

$$\begin{aligned}
\min \quad & \sum_{r\in V}\sum_{e\in E} c_e \cdot \left(\max_{u\in\mathcal{T}} f_{r,u,e}\right) \;+\; \sum_{r\in V}\sum_{e\in E} \ell_e \cdot \sum_{u\in\mathcal{T}} f_{r,u,e} \\
\text{s.t.} \quad & \sum_{r\in R_s} f_{r,s_i}(S_r) \;+\; \sum_{r\in R_t} f_{r,t_i}(T_r) \geq 1, \qquad \forall i \in [m],\; \forall (R_s, R_t) \text{ partition of } V, \\
& \qquad \forall S_r : s_i - r \text{ cut}, \forall r \in R_s,\; \forall T_r : r - t_i \text{ cut}, \forall r \in R_t \\
& f \geq 0.
\end{aligned}$$

Note that $\ell_{\log(n)}$-norm is a constant approximation for ℓ_∞. Therefore we can reformulate the above objective function (at the loss of a constant factor) as the sum of $\ell_{\log(n)}$ and ℓ_1 norms. Our fractional solver applies to this convex covering problem, and yields an $O(\log n)$-competitive ratio (note that $\rho = 1$ for this instance). In order to get a polynomial running time, we can use the natural "separation oracle" approach to produce violated covering constraints.

Each iteration above runs in polynomial time since the minimum cuts can be computed in polynomial time. In order to bound the number of iterations, consider the potential $\psi = \sum_{e\in E}(f_{r,s_i,e} + f_{r,t_i,e})$. Note that $0 \leq \psi \leq 2|E|$ and each iteration increases ψ by at least $\frac{1}{2}$. So the number of iterations is at most $4|E|$.

When the i^{th} request (s_i, t_i) arrives
repeat
  For each $r \in V$, compute $\mathsf{MC}(r, s_i)$ and $\mathsf{MC}(r, t_i)$ and the respective cuts S_r and T_r;
  Let $R_s = \{r \in V : \mathsf{MC}(r, s_i) \leq \mathsf{MC}(r, t_i)\}$ and $R_t = V \setminus R_s$;
  Run Algorithm 1 with constraint $\sum_{r \in R_s} f_{r,s_i}(S_r) + \sum_{r \in R_t} f_{r,t_i}(T_r) \geq 1$;
until $\sum_{r \in V} \min\{\mathsf{MC}(r, s_i), \mathsf{MC}(r, t_i)\} \geq \frac{1}{2}$;

Algorithm 2: Separation Oracle Based Algorithm for Buy-at-Bulk.

4.2 Throughput Maximization with ℓ_p-norm Capacities

The online problem of maximizing multicommodity flow was studied in [4, 13]. In this problem, we are given a directed graph with edge capacities $u(e)$. Requests (s_i, t_i) arrive in an online fashion. The algorithm should choose a path between s_i and t_i and allocate a bandwidth of 1 on the path to serve request i. The total bandwidth allocated on any edge is not allowed to exceed its capacity. This is the simplest version of the multicommodity routing problem. Here we consider an extension with ℓ_p-norm capacity constraints on subsets of edges. This can be used to model situations where edges are provided by multiple agents. Each agent j owns a subset S_j of edges and it requires the ℓ_{p_j}-norm of the bandwidths of these edges to be at most c_j. In this section we assume the S_j are disjoint. Our result also applies to general S_j via a reduction to disjoint instances.

▶ **Theorem 7.** *Assume that $c_j = \Omega(\log m) \cdot |S_j|^{1/p_j}$ for each j. Then there is a randomized $O(\log m)$-competitive online algorithm for throughput maximization with ℓ_p-norm capacities, where m is the number of edges in the graph.*

We note that a similar "high capacity" assumption is also needed in the linear special case [4, 13] where each $|S_j| = 1$.

In a fractional version of the problem, a request can be satisfied by several paths and the allocation of bandwidth can be in range $[0, 1]$ instead of being restricted from $\{0, 1\}$. For request (s_i, t_i), let $\mathcal{P}_i$ be the set of simple paths between s_i and t_i. Variable $f_{i,P}$ is defined to be the amount of flow on the path P for request (s_i, t_i). The total profit of algorithm is the (fractional) number of requests served and the performance is measured with respect to the maximum number of requests that could be served if the requests are known beforehand. We describe the problem as a packing problem:

$$\max \quad \sum_i \sum_{P \in \mathcal{P}_i} f_{i,P} \tag{6}$$

$$\text{s.t.} \quad \sum_{P \in \mathcal{P}_i} f_{i,P} \leq 1, \qquad \forall i \tag{7}$$

$$\sum_i \sum_{P \in \mathcal{P}_i : e \in P} f_{i,P} = \mu_e, \qquad \forall e \tag{8}$$

$$\|\mu(S_j)\|_{p_j} \leq c_j, \qquad \forall j \tag{9}$$

$$f \in \mathbb{R}^n_+$$

Note that single edge capacity is a special case of (9) with $|S_j| = 1$ and any p_j. The corresponding primal problem is the following.

$$\min \quad \sum_j c_j \|x(S_j)\|_{q_j} + \sum_i z_i \tag{10}$$

$$\text{s.t.} \quad z_i + \sum_{e \in P} x_e \geq 1, \qquad \forall i,\ P \in \mathcal{P}_i \tag{11}$$

$$x, z \in \mathbb{R}^n_+$$

where z is the dual variable of constraints (7) and x is the dual variable of constraints (8) to (9). This formulation falls in the form of (1). Combining our algorithm with online rounding techniques, we can prove Theorem 7. A formal proof appears in the full version.

References

1. Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.
2. Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009.
3. Baruch Awerbuch, Yossi Azar, Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Load balancing in the l_p norm. In *FOCS*, pages 383–391, 1995.
4. Baruch Awerbuch, Yossi Azar, and Serge Plotkin. Throughput-competitive on-line routing. In *FOCS*, pages 32–40. IEEE, 1993.
5. Yossi Azar, Umang Bhaskar, Lisa K. Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *SODA*, 2013.
6. Yossi Azar, Niv Buchbinder, T.-H. Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Naor, and Debmalya Panigrahi. Online algorithms for covering and packing problems with convex objectives. In *FOCS*, pages 148–157, 2016.
7. Yossi Azar, Ilan Reuven Cohen, and Debmalya Panigrahi. Online covering with convex objectives and applications. *CoRR*, abs/1412.3507, 2014. URL: `http://arxiv.org/abs/1412.3507`.
8. Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. *SIAM J. Comput.*, 41(2):391–414, 2012.
9. Nikhil Bansal and Kirk Pruhs. Server scheduling to balance priorities, fairness, and average quality of service. *SIAM J. Comput.*, 39(7):3311–3335, 2010.
10. Avrim Blum, Anupam Gupta, Yishay Mansour, and Ankit Sharma. Welfare and profit maximization with production costs. In *FOCS*, pages 77–86, 2011.
11. Niv Buchbinder, Shahar Chen, Anupam Gupta, Viswanath Nagarajan, and Joseph Naor. Online packing and covering framework with convex objectives. *CoRR*, abs/1412.8347, 2014.
12. Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264, 2007.
13. Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009.
14. Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal-dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2007.
15. T.-H. Hubert Chan, Zhiyi Huang, and Ning Kang. Online convex covering and packing problems. *CoRR*, abs/1502.01802, 2015.

16 Moses Charikar and Adriana Karagiozova. On non-uniform multicommodity buy-at-bulk network design. In *STOC*, pages 176–182. ACM, 2005.

17 Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. Approximation algorithms for nonuniform buy-at-bulk network design. *SIAM J. Comput.*, 39(5):1772–1798, 2010.

18 Nikhil R. Devanur and Zhiyi Huang. Primal dual gives almost optimal energy efficient online algorithms. In *SODA*, pages 1123–1140, 2014.

19 Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *STOC*, pages 137–144. ACM, 2012.

20 Noa Elad, Satyen Kale, and Joseph (Seffi) Naor. Online semidefinite programming. In *ICALP*, pages 40:1–40:13, 2016.

21 Alina Ene, Deeparnab Chakrabarty, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. In *FOCS*, pages 545–562, 2015.

22 Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *WAOA*, pages 173–186, 2012.

23 Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. *Math. Oper. Res.*, 39(4):998–1011, 2014.

24 Zhiyi Huang and Anthony Kim. Welfare maximization with production costs: A primal dual approach. In *SODA*, pages 59–72, 2015.

25 Ishai Menache and Mohit Singh. Online caching with convex costs: Extended abstract. In *SPAA*, pages 46–54, 2015.

Dynamic Beats Fixed: On Phase-Based Algorithms for File Migration*†

Marcin Bienkowski[1], Jarosław Byrka[2], and Marcin Mucha[3]

1 **Institute of Computer Science, University of Wrocław, Wrocław, Poland**
`mbi@cs.uni.wroc.pl`
2 **Institute of Computer Science, University of Wrocław, Wrocław, Poland**
`jby@cs.uni.wroc.pl`
3 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
`mucha@mimuw.edu.pl`

Abstract

In this paper, we construct a deterministic 4-competitive algorithm for the online file migration problem, beating the currently best 20-year old, 4.086-competitive MTLM algorithm by Bartal et al. (SODA 1997). Like MTLM, our algorithm also operates in phases, but it adapts their lengths dynamically depending on the geometry of requests seen so far. The improvement was obtained by carefully analyzing a linear model (factor-revealing LP) of a single phase of the algorithm. We also show that if an online algorithm operates in phases of fixed length and the adversary is able to modify the graph between phases, no algorithm can beat the competitive ratio of 4.086.

1998 ACM Subject Classification F.1.2 [Modes of Computation] Online Computation, G.1.6 [Optimization] Linear programming, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases file migration, factor-revealing linear programs, online algorithms, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.13

1 Introduction

Consider the problem of managing a shared data item among sets of processors. For example, in a distributed program running in a network, nodes want to have access to shared files, objects or databases. Such a file can be stored in the local memory of one of the processors and when another processor wants to access (read from or write to) this file, it has to contact the processor holding the file. Such a transaction incurs a certain cost. Moreover, access patterns to this file may change frequently and unpredictably, which renders any static placement of the file inefficient. Hence, the goal is to minimize the total cost of communication by moving the file in response to such accesses, so that the requesting processors find the file "nearby" in the network.

The *file migration* problem serves as the theoretical underpinning of the application scenario described above. The problem was coined by Black and Sleator [13] and was initially called *page migration*, as the original motivation concerned managing a set of memory pages

* Extended abstract; the full version is available at `https://arxiv.org/abs/1609.00831`.

† Partially supported by Polish National Science Centre grants 2016/22/E/ST6/00499 and 2015/18/E/ST6/00456. The work of M. Mucha is part of a project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 677651).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 13; pp. 13:1–13:14

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

in a multiprocessor system. There the data item was a single memory page held at a local memory of a single processor.

Most subsequent work referred to this problem as *file migration* and we will stick to this convention in this paper. The file migration problem assumes the *non-uniform model*, where the shared file is much larger than a portion accessed in a single time step. This is typical when in one step a processor wants to read a single unit of data from a file or a record from a database. On the other hand, to reduce the maintenance overhead, it is assumed that the shared file is indivisible, and can be migrated between nodes only as a whole. This makes the file migration much more expensive than a single access to the file. As the knowledge of future accesses is either partial or completely non-existing, the accesses to the file can be naturally modeled as an online problem, where the input sequence consists of processor identifiers, which sequentially try to access pieces of the shared file.

1.1 The Model

The studied network is modeled as an edge-weighted graph or, more generally, as a metric space $(\mathcal{X}, d)$ whose point set $\mathcal{X}$ corresponds to processors and d defines the distances between them. There is a large indivisible *file* (historically called *page*) of size D stored at a point of $\mathcal{X}$. An input is a sequence of space points $r_1, r_2, r_3, \ldots$ denoting processors requesting access to the file. This sequence is presented in an online manner to an algorithm. More precisely, we assume that the time is slotted into steps numbered from 1. Let ALG_t denote the position of the file at the end of step t and ALG_0 be the initial position of the file. In step $t \geq 1$, the following happens:

1. A requesting point r_t is presented to the algorithm.
2. The algorithm pays $d(\text{ALG}_{t-1}, r_t)$ for serving the request.
3. The algorithm chooses a new position ALG_t for the file (possibly $\text{ALG}_t = \text{ALG}_{t-1}$) and moves the file to ALG_t paying $D \cdot d(\text{ALG}_{t-1}, \text{ALG}_t)$.

After the t-th request, the algorithm has to make its decision (where to migrate the file) exclusively on the basis of the sequence up to step t. To measure the performance of an online strategy, we use the standard competitive ratio metric [14]: an online deterministic algorithm Alg is *c-competitive* if there exists a constant γ, such that for any input sequence $\mathcal{I}$, it holds that $C_{\text{ALG}}(\mathcal{I}) \leq c \cdot C_{\text{OPT}}(\mathcal{I}) + \gamma$, where C_{ALG} and C_{OPT} denote the costs of Alg and Opt (optimal *offline* algorithm) on $\mathcal{I}$, respectively. The minimum c for which Alg is c-competitive is called the *competitive ratio* of Alg.

1.2 Previous Work

The problem was stated by Black and Sleator [13], who gave 3-competitive deterministic algorithms for uniform metrics and trees and conjectured that 3-competitive deterministic algorithms were possible for any metric space.

Westbrook [26] constructed randomized strategies: a 3-competitive algorithm against adaptive-online adversaries and a $(1 + \phi)$-competitive algorithm (for D tending to infinity) against oblivious adversaries, where $\phi \approx 1.618$ denotes the golden ratio. By the result of Ben-David et al. [10] this asserted *the existence* of a deterministic algorithm with the competitive ratio at most $3 \cdot (1 + \phi) \approx 7.854$.

The first explicit deterministic construction was the 7-competitive algorithm Move-To-Min (Mtm) by Awerbuch et al. [2]. Mtm operates in phases of length D, during which the algorithm *remains at a fixed position*. In the last step of a phase, Mtm migrates the file to

a point that minimizes the sum of distances to all requests $r_1, r_2, \ldots, r_D$ presented in the phase, i.e., to a minimizer of the function $f_{\text{MTM}}(x) = \sum_{i=1}^{D} d(x, r_i)$.

The ratio has been subsequently improved by the algorithm Move-To-Local-Min (Mtlm) by Bartal et al. [8]. Mtlm works similarly to Mtm, but it changes the phase duration to $c_0 \cdot D$ for a constant c_0, and when computing the new position for the file, it also takes the migration distance into consideration. Namely, it chooses to migrate the file to a point that minimizes the function

$$f_{\text{MTLM}}(x) = D \cdot d(v_{\text{MTLM}}, x) + \tfrac{c_0+1}{c_0} \sum_{i=1}^{c_0 \cdot D} d(x, r_i) \ ,$$

where v_{MTLM} denotes the point at which Mtlm keeps its file during the phase. The algorithm is optimized by setting $c_0 \approx 1.841$ being the only positive root of the equation $3c^3 - 8c - 4 = 0$. For such c, the competitive ratio of Mtlm is $R_0 \approx 4.086$, where R_0 is the largest (real) root of the equation $R^3 - 5R^2 + 3R + 3 = 0$. Their analysis is tight.

It is worth noting that most of the competitive ratios given above hold when D tends to infinity. In particular, for Mtlm we assume that $c_0 \cdot D$ is an integer and the ratio of $1 + \phi$ of Westbrook's algorithm [26] is achieved only in the limit.

Better deterministic algorithms are known only for some specific graph topologies. There are 3-competitive algorithms for uniform metrics and trees [13], and $(3 + 1/D)$-competitive strategies for three-point metrics [23]. Chrobak et al. [15] showed $2 + 1/(2D)$-competitive strategies for continuous trees and products of trees, e.g., for $\mathbb{R}^n$ with ℓ_1 norm. Furthermore, a $(1 + \phi)$-competitive algorithm for $\mathbb{R}^n$ under any norm was also given in [15].

A straightforward lower bound of 3 for deterministic algorithms was given by Black and Sleator [13] and later adapted to randomized algorithms against adaptive-online adversaries by Westbrook [26]. The currently best lower bound for deterministic algorithms is due to Matsubayashi [22], who showed a lower bound of $3 + \varepsilon$ that holds for any value of D, where ε is a constant that does not depend on D. This renders the file migration problem one of the few natural problems, where a known lower bound on the competitive ratio of any deterministic algorithm is strictly larger than the competitive ratio of a randomized algorithm against an adaptive-online adversary.

Finally, improved results were given for a simplified model where $D = 1$: the competitive ratio for deterministic algorithms is then known to be between 3.164 and 3.414 [21].

1.3 Our Contribution

We propose a new deterministic algorithm that dynamically decides on the length of the phase based on the geometry of requests received in the initial part of each phase. This improves the 20-years old result of Bartal et al. [8].

The improvement was obtained by carefully analyzing a linear model (factor revealing LP) of a single phase of the algorithm. It allowed us to identify some key tight examples for the previous analysis, suggested a nontrivial construction of the new algorithm, and facilitated a systematic search within the design parameter space.

More precisely, for a fixed algorithm Alg (from a relatively broad class), we create a maximization LP with the following property: if the competitive ratio of Alg is at least R, then so is the value of LP. A solution to the LP contains a succinct description of a metric space along with a short description of a single-phase input, both constituting a lower bound for Alg. Hence, the value of LP is an upper bound on Alg's competitiveness. We discuss the details of the LP approach in Section 4.

The way the algorithm was obtained is perhaps unintuitive. Nevertheless, the final algorithm is an elegant construction involving only essentially integral constants. By studying

the dual solution, we managed to extract a compact, human-readable, combinatorial upper bound based on path-packing arguments and to obtain the following result.

▶ **Theorem 1.** *There exists a deterministic 4-competitive algorithm for the file migration problem.*

We also show that improvement of MTLM would not be possible by just selecting different parameters for a phase-based algorithm operating with a fixed phase length. Our construction shows that an analysis that treats each phase separately (e.g., the one employed for MTLM [8]) cannot give better bounds on the competitive ratio than 4.086. (A weaker lower bound of 3.847 for algorithms that use fixed phase length was given by Bartal et al. [8].)

▶ **Theorem 2.** *Fix any algorithm* ALG *that operates in phases of fixed length. Assume that between the phases, the adversary can arbitrarily modify the graph while keeping the distance between the files of* OPT *and* ALG *unchanged. Then, the competitive ratio of* ALG *is at least R_0 (for D tending to infinity), where $R_0 \approx 4.086$ is the competitive ratio of algorithm* MTLM*.*

1.4 Other Related Work

The file migration problem has been generalized in a few directions. When we lift the restriction that the file can only be migrated and not copied, the resulting problem is called *file allocation* [9, 2, 18]. It makes sense especially when we differentiate read and write requests to the file; for the former, we need to contact only one replica of the file; for the latter all copies need to be updated. The attainable competitive ratios become then worse: the best deterministic algorithm is $O(\log n)$-competitive [2]; the lower bound of $\Omega(\log n)$ holds even for randomized algorithms and follows by a reduction from the online Steiner tree problem [9, 17].

The file migration problem has been also extended to accommodate memory capacity constraints at nodes (when more than one file is used) [1, 3, 4, 6], dynamically changing networks [4, 12], and different objective functions (e.g., minimizing congestion) [19, 25]. For a more systematic treatment of the file migration and related problems, see surveys [7, 11]. For more applied approaches, see the survey [16] and the references therein.

2 4-Competitive Algorithm Dynamic-Local-Min

We start with an insight concerning the hard inputs for the MTLM algorithm [8]. We identified two classes of tight instances for MTLM: *bipartite* and *linear* (cf. Figure 1). It can be shown that if the algorithm knew in advance on which instance it was run, it could improve its performance by changing the phase length. Namely, for bipartite instances a longer phase would help the algorithm, whereas a shorter phase would be beneficial for linear instances.

To decide the length of the phase, we need to measure the level of request concentration as compared to the distance from the current position of an algorithm to the center of requests. Intuitively, observing that (from some time) requests are concentrated around a certain point motivates the algorithm to shorten the phase and quickly move to the "center of the requests". If, on the other hand, requests are scattered and the current algorithm's position is essentially in the middle of the observed requests, it appears reasonable to wait longer before moving the file. This rule agrees with the desired behavior of the algorithm on linear and bipartite instances.

Turning the above intuition into an effective phase extension rule is not trivial. We present an algorithm based on a rule that we have extracted from an optimization process

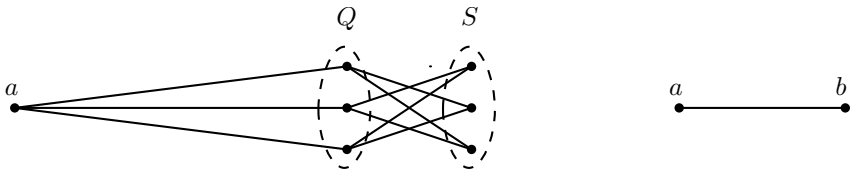


Figure 1 The geometries of selected tight instances for MTLM. In both cases, the algorithm starts at point a. In the *linear* instance (on the right), the requests are initially given at a and then later at b, and the algorithm is expected to migrate the file to point b. In the *bipartite* instance (on the left) the requests are given at nodes from set S and the algorithm is expected to migrate the file to one of the nodes from set Q.

using a natural linear model of the amortized phase-based analysis. This linear model is quite complex and we present it in Section 4. It can be seen as an alternative (computer-based) proof for the performance guarantee of our algorithm. Such proof technique might be interesting on its own and useful for analyzing other online games played on metric spaces.

2.1 Notation

For succinctness, we introduce the following notions. For any two points $v_1, v_2 \in \mathcal{X}$, let $[v_1, v_2] = D \cdot d(v_1, v_2)$. We extend this notation to sequences of points, i.e., $[v_1, v_2, \ldots, v_j] = [v_1, v_2] + [v_2, v_3] + \ldots + [v_{j-1}, v_j]$. Moreover, if $v \in \mathcal{X}$ is a point and $S \subseteq \mathcal{X}$ is a multiset of points, then

$$[v, S] = [S, v] = D \cdot \tfrac{1}{|S|} \textstyle\sum_{x \in S} d(v, x) \ ,$$

i.e., $[v, S]$ is the average distance from v to a point of S times D. We extend the sequence notation introduced above to sequences of points and multisets of points, e.g., $[v, S, u, T] = [v, S] + [S, u] + [u, T]$. The symbol $[S, T]$ is not defined for multisets S, T; we will only use this notation for sequences that do not contain two consecutive multisets.

Observe that the sequence notation allows for easy expressing of the triangle inequality: $[v_1, v_2] \leq [v_1, v_3, v_2]$; we will extensively use this property. Note that the following "multiset" version of the triangle inequality also holds: $[v_1, v_2] \leq [v_1, S, v_2]$.

2.2 Algorithm definition

We propose a new phase-based algorithm that dynamically decides on the length of the current phase, which we call Dynamic-Local-Min (DLM). DLM operates in phases, but it chooses their lengths depending on the geometry of requests seen in the initial part of the phase. Roughly speaking, when it recognizes that the currently seen requests more closely resemble a linear tight example for MTLM, it ends the phase after $1.75\,D$ steps. Otherwise, it assumes that the presented graph is more in the flavor of the bipartite construction, and ends the phase only after $2.25\,D$ steps.

For any step t, we denote the position of DLM's file at the end of step t by DLM_t and that of OPT by OP_t. We identify the requests with the points where they are issued.

Assume a phase starts in step $t+1$; that is, DLM_t is the position of DLM at the very beginning of a phase. Within the phase, DLM waits $1.75\,D$ steps and at step $t + 1.75\,D$, it finds a node v_g that minimizes the function

$$g(v) = [\text{DLM}_t, v, \mathcal{R}_1, v, \mathcal{R}_2] = [\text{DLM}_t, v] + 2 \cdot [v, \mathcal{R}_1] + [v, \mathcal{R}_2] \ ,$$

where $\mathcal{R}_1$ is the multiset of the requests from steps $t+1,\ldots,t+D$ and $\mathcal{R}_2$ is the multiset of the subsequent requests from steps $t+D+1,\ldots,t+1.75\,D$.

If $g(v_g) \le 1.5 \cdot [\textsc{dlm}_t, \mathcal{R}_2]$, the algorithm moves its file to v_g, and ends the current phase. Intuitively, this condition corresponds to detecting if there exists a point that is substantially closer to the first $1.75\,D$ requests of the phase than the current position. The condition is constructed in a way to be conclusive for each of the possible outcomes. If indeed such point exists, then by moving the file to this point, we get much closer to the file of Opt. If there is no such good point, then also the optimal solution is experiencing some request related costs. Then, we may afford to wait a little longer and meanwhile get a more accurate estimation of the possible location of the file of Opt.

That is, if $g(v_g) > 1.5 \cdot [\textsc{dlm}_t, \mathcal{R}_2]$, Dlm waits the next $0.5\,D$ steps and (in step $t+2.25\,D$) it moves its file to the point v_h, where v_h is the minimizer of the function

$$h(v) = [\textsc{dlm}_t, v] + [v, \mathcal{R}_1] + 1.25 \cdot [v, \mathcal{R}_2] + 0.75 \cdot [v, \mathcal{R}_3] \ .$$

$\mathcal{R}_3$ is the multiset of the last $0.5\,D$ requests from the prolonged phase (from steps $t+1.75\,D+1,\ldots,t+2.25\,D$). Also in this case, the next phase starts right after the file movement.

Note that the *short phase* consists of D requests denoted $\mathcal{R}_1$ followed by $0.75\,D$ requests denoted $\mathcal{R}_2$, while the *long phase* consists additionally of $0.5\,D$ requests denoted $\mathcal{R}_3$. We will say that the short phase consists of *two parts*, $\mathcal{R}_1$ and $\mathcal{R}_2$, and the long phase consists of *three parts*, $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$.

2.3 DLM Analysis

We start with a lower bound on Opt. The following bound is an extension of the bound given implicitly in [8]; its proof is given in the full version of the paper.

▶ **Lemma 3.** *Let $\mathcal{R}$ be a subsequence of at most $2D$ consecutive requests from the input issued at steps $t+1, t+2, \ldots, t+|\mathcal{R}|$. Then, $4 \cdot C_{\mathrm{OPT}}(\mathcal{R}) \ge (2|\mathcal{R}|/D) \cdot [\textsc{op}_t, \mathcal{R}, \textsc{op}_{t+|\mathcal{R}|}] + (4 - 2|\mathcal{R}|/D) \cdot [\textsc{op}_t, \textsc{op}_{t+|\mathcal{R}|}]$.*

We define a potential function at (the end of) step t as $\Phi_t = 3 \cdot [\textsc{dlm}_t, \textsc{op}_t]$. It suffices to show that in any (short or long) phase consisting of steps $t+1, t+2, \ldots, t+\ell$, during which requests $\mathcal{R}$ are given, it holds that

$$C_{\mathrm{ALG}}(\mathcal{R}) + \Phi_{t+\ell} \le 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}) + \Phi_t \ . \tag{1}$$

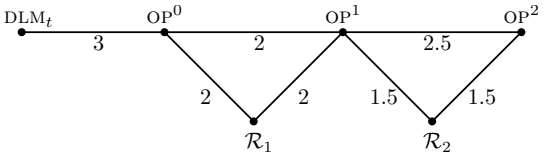
Theorem 1 follows immediately by summing the above bound over all phases of the input.

2.3.1 Proof for a short phase

We consider any short phase $\mathcal{R}$ consisting of part $\mathcal{R}_1$, spanning steps $t+1,\ldots,t+D$, and part $\mathcal{R}_2$, spanning steps $t+D+1,\ldots,t+1.75\,D$. For succinctness, we define $\textsc{op}^0 = \textsc{op}_t$, $\textsc{op}^1 = \textsc{op}_{t+D}$ and $\textsc{op}^2 = \textsc{op}_{t+1.75\,D}$. By Lemma 3 applied to $\mathcal{R}_1$ and $\mathcal{R}_2$,

$$\begin{aligned} 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}) + \Phi_t \ &= \ 3 \cdot [\textsc{dlm}_t, \textsc{op}^0] + 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}_1) + 4 \cdot C_{\mathrm{OPT}}(\mathcal{R}_2) \\ &\ge \ 3 \cdot [\textsc{dlm}_t, \textsc{op}^0] + 2 \cdot [\textsc{op}^0, \textsc{op}^1] + 2 \cdot [\textsc{op}^0, \mathcal{R}_1, \textsc{op}^1] \\ &\quad + 2.5 \cdot [\textsc{op}^1, \textsc{op}^2] + 1.5 \cdot [\textsc{op}^1, \mathcal{R}_2, \textsc{op}^2] \ . \end{aligned} \tag{2}$$

We treat the amount (2) as our budget. This is illustrated below; the coefficients are written as edge weights.



Budget = 4 OPT
+ initial potential
(short phase case)

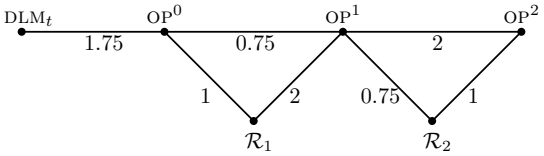
Now, we bound $C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+1.75\,D}$ using the definition of ALG and the triangle inequality.

$$\begin{aligned}
&C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+1.75\,D} \\
&\quad = C_{\text{ALG}}(\mathcal{R}_1) + C_{\text{ALG}}(\mathcal{R}_2) + 3[v_g, \text{OP}^2] \\
&\quad \le [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + [\text{DLM}_t, v_g] + 3 \cdot [v_g, \text{OP}^2] \\
&\quad \le [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + [\text{DLM}_t, v_g] + 2 \cdot [v_g, \mathcal{R}_1, \text{OP}^2] + [v_g, \mathcal{R}_2, \text{OP}^2] \\
&\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] \\
&\qquad\quad + [\text{DLM}_t, v_g] + 2 \cdot [v_g, \mathcal{R}_1] + [v_g, \mathcal{R}_2] \\
&\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] + g(v_g)\ . \qquad (3)
\end{aligned}$$

The first four summands of (3) can be bounded as

$$\begin{aligned}
&[\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] \qquad (4) \\
&\quad \le [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \text{OP}^1, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2]\ ,
\end{aligned}$$

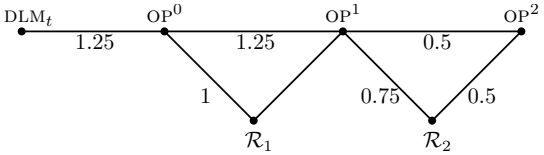
and their total weights in the final expression are depicted below.



DLM cost
+ final potential
(part 1, short phase)

For the last summand of (3), $g(v_g)$, we use the fact that v_g is a minimizer of the function g (and hence $g(v_g) \le g(\text{OP}^0)$), and the property of the short phase ($g(v_g) \le 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$). Therefore,

$$\begin{aligned}
g(v_g) \ &\le\ 0.5 \cdot g(\text{OP}^0) + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \\
&\le\ 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \mathcal{R}_2] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \qquad (5) \\
&\le\ 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \text{OP}^2, \mathcal{R}_2] + 0.75 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \mathcal{R}_2]\ .
\end{aligned}$$



DLM cost
+ final potential
(part 2, short phase)

By combining (3), (4) and (5) (or simply adding the edge coefficients on the last two figures) we observe that the budget ((2), i.e., the edge coefficients on the first figure) is not exceeded. This implies 4-competitiveness, i.e., that (1) holds for any short phase.

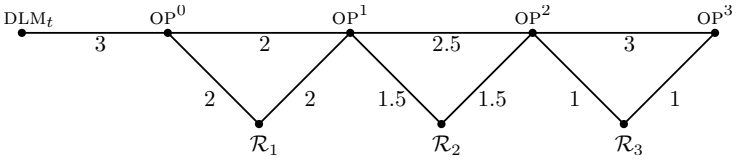
2.3.2 Proof for a long phase

We consider any long phase $\mathcal{R}$ consisting of part $\mathcal{R}_1$, spanning steps $t+1, \ldots, t+D$; part $\mathcal{R}_2$, spanning steps $t+D+1, \ldots, t+1.75 \cdot D$; and part $\mathcal{R}_3$, spanning steps $t+1.75 \cdot D+1, \ldots, t+2.25 \cdot D$. Similarly to the proof for a short phase, we define $\text{OP}^0 = \text{OP}_t$, $\text{OP}^1 = \text{OP}_{t+D}$,

$\text{OP}^2 = \text{OP}_{t+1.75\,D}$, and $\text{OP}^3 = \text{OP}_{t+2.25\,D}$. We emphasize that the positions of Opt in a long and a short phase can be completely different.

By Lemma 3, we obtain a bound very similar to that for a short phase; again, we treat it as a budget and depict its coefficients as edge weights.

$$\begin{aligned} 4 \cdot C_{\text{OPT}}(\mathcal{R}) + \Phi_t &= 3 \cdot [\text{DLM}_t, \text{OP}^0] + 4 \cdot C_{\text{OPT}}(\mathcal{R}_1) + 4 \cdot C_{\text{OPT}}(\mathcal{R}_2) + 4 \cdot C_{\text{OPT}}(\mathcal{R}_3) \\ &\geq 3 \cdot [\text{DLM}_t, \text{OP}^0] + 2 \cdot [\text{OP}^0, \text{OP}^1] + 2 \cdot [\text{OP}^0, \mathcal{R}_1, \text{OP}^1] \\ &\quad + 2.5 \cdot [\text{OP}^1, \text{OP}^2] + 1.5 \cdot [\text{OP}^1, \mathcal{R}_2, \text{OP}^2] \\ &\quad + 3 \cdot [\text{OP}^2, \text{OP}^3] + [\text{OP}^2, \mathcal{R}_3, \text{OP}^2] \ . \end{aligned} \tag{6}$$



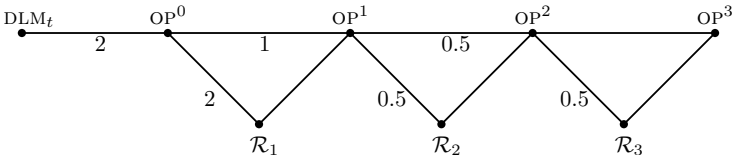
Budget = 4 OPT + initial potential (long phase case)

Now, we bound $C_{\text{ALG}}(R) + \Phi_{t+2.25\,D}$, using the definition of Alg and the triangle inequality.

$$\begin{aligned} &C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+2.25\,D} \\ &\quad = C_{\text{ALG}}(\mathcal{R}_1) + C_{\text{ALG}}(\mathcal{R}_2) + C_{\text{ALG}}(\mathcal{R}_3) + 3 \cdot [v_h, \text{OP}^3] \\ &\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] + [\text{DLM}_t, v_h] + 3 \cdot [v_h, \text{OP}^3] \\ &\quad \leq [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] + [\text{DLM}_t, v_h] \\ &\qquad + [v_h, \mathcal{R}_1, \text{OP}^3] + 1.25 \cdot [v_h, \mathcal{R}_2, \text{OP}^3] + 0.75 \cdot [v_h, \mathcal{R}_3, \text{OP}^3] \\ &\quad = [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] \\ &\qquad + [\text{OP}^3, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] + h(v_h) \ . \end{aligned} \tag{7}$$

Since Dlm has not migrated the file after the first two parts, $g(v) \geq 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$ for any node v. Therefore $0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \leq 0.5 \cdot g(\text{OP}^0) = 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \mathcal{R}_2] \leq 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \mathcal{R}_2]$. Using this and the triangle inequality, the first three summands of (7) can be bounded and depicted as follows:

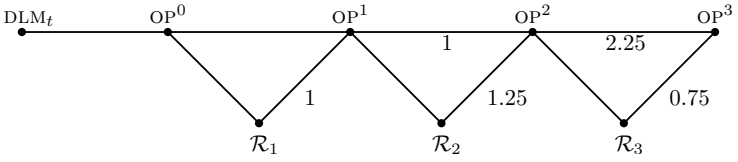
$$\begin{aligned} &[\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] \\ &\quad \leq [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1] + 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \mathcal{R}_2] \\ &\qquad + 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \text{OP}^2, \mathcal{R}_3] \ . \end{aligned} \tag{8}$$



DLM cost + final potential (part1, long phase)

The next three summands of (7) can be also bounded appropriately:

$$\begin{aligned} &[\text{OP}^3, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] \\ &\quad \leq [\text{OP}^3, \text{OP}^2, \text{OP}^1, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \text{OP}^2, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] \ . \end{aligned} \tag{9}$$

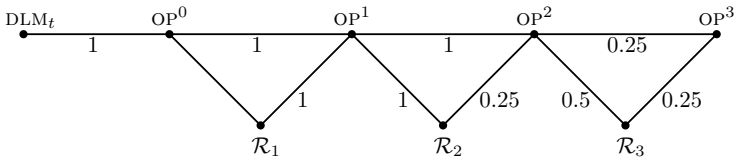


DLM cost + final potential (part 2, long phase)

Lastly, for bounding $h(v_h)$, we use the fact that v_h is a minimizer of h, and hence

$$\begin{aligned} h(v_h) &\leq h(\text{OP}^1) \\ &= [\text{OP}^1, \text{DLM}_t] + [\text{OP}^1, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^1, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^1, \mathcal{R}_3] \\ &\leq [\text{OP}^1, \text{OP}^0, \text{DLM}_t] + [\text{OP}^1, \mathcal{R}_1] + [\text{OP}^1, \mathcal{R}^2] + 0.25 \cdot [\text{OP}^1, \text{OP}^2, \mathcal{R}_2] \\ &\qquad + 0.5 \cdot [\text{OP}^1, \text{OP}^2, \mathcal{R}_3] + 0.25 \cdot [\text{OP}^1, \text{OP}^2, \text{OP}^3, \mathcal{R}_3] \ . \end{aligned} \tag{10}$$

Note that in (10) we split some of the paths and choose the longer ones, so that the budgets on edges are not violated. Bound (10) is depicted on the figure below.



DLM cost
+ final potential
(part 3, long phase)

By combining (7), (8), (9) and (10) (or simply adding edge coefficients on the last three figures), we observe that the budget ((6), i.e., the edge coefficients on the first figure) is not exceeded. This implies 4-competitiveness, i.e., that (1) holds for any long phase and concludes the proof of Theorem 1.

3 Lower Bound for Phase-Based Algorithms

A *fixed-phase algorithm* chooses phase length $c \cdot D$ and after every $c \cdot D$ requests it makes a migration decision solely on the basis of its current position and the last $c \cdot D$ requests. We now proceed to argue that no fixed-phase algorithm ALG can beat the competitive ratio $R_0 \approx 4.086$ achieved by MTLM [8] (cf. Theorem 2). As already stated in the introduction, to ensure that the algorithm cannot base its choices on the previous phases, we will give the adversary an additional power: it may modify the graph between the phases of ALG, as long as the distance between nodes keeping the files of ALG and OPT (v_{ALG} and v_{OPT}, respectively) is preserved. We emphasize that the analysis of MTLM [8] essentially uses this model: each phase is analyzed completely separately from others. The full proof is given in the full version of the paper; here we informally highlight its key ideas.

The adversarial construction consists of many epochs, each consisting of some number of phases. At the beginning and at the end of an epoch, ALG and OPT keep their files at the same node. We define three adversarial strategies, called *plays*: *linear*, *bipartite*, and *finishing*. Each play consists of one or more phases. A prerequisite for each given play is a particular distance between v_{ALG} and v_{OPT}. Each play will have some properties: it will incur some cost on ALG and OPT and will end with v_{ALG} and v_{OPT} in a specific distance.

In the first phase of an epoch, when initially $v_{\text{ALG}} = v_{\text{OPT}}$, the adversary uses the *linear* play (the generated graph is a single edge of length 1), so that at the end of the phase, $d(v_{\text{ALG}}, v_{\text{OPT}}) = 1$. For such phase P, we have $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P) - (1/(1-2\alpha)) \cdot D$, where $\alpha = 1/(R_0 - 1)$. Note that in this phase alone, the adversary does not enforce the desired competitive ratio of R_0, but it increases the distance between v_{OPT} and v_{ALG}.

In each of the next L phases, the adversary employs the *bipartite* play; the graph used corresponds to a tight bipartite example for MTLM, cf. Figure 1). Let f be the value of $d(v_{\text{ALG}}, v_{\text{OPT}})$ at the beginning of a phase. If the algorithm plays well, then at the end of the phase this distance decreases to $2\alpha \cdot f$. Furthermore, neglecting lower order terms, for such phase P, it holds that $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P) + f \cdot D$, i.e., the inequality $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P)$ holds with the slack $f \cdot D$. The sum of these slacks over L phases

is $\sum_{i=0}^{L-1}(2\alpha)^i \cdot D$, which tends to $(1/(1-2\alpha)) \cdot D$ when L grows. Hence, after one linear and L bipartite phases (for a large L and neglecting lower order terms again), the cost paid by Alg is at least R_0 times the cost paid by Opt and the distance between their files is negligible.

Finally, to decrease the distance between v_{ALG} and v_{OPT} to zero, the adversary uses a third type of play, the *finishing* one. This play incurs a negligible cost and it forces the positions of Alg and Opt files to coincide, which ends an epoch.

4 Linear Program for File Migration

In this section, we present a linear programming model for the analysis of both algorithm Mtlm by Bartal et al. [8] and our algorithm Dlm.

4.1 LP analysis of MTLM-like algorithms

In our approach, we analyze any Mtlm-like algorithm Alg. We use our notion of distances from Section 2.1. Alg will be a variant of Mtlm parameterized with two values β and δ. The length of its phase is $\delta \cdot D$ and the initial point of Alg is denoted by A_0. We denote the set of requests within a phase by $\mathcal{R}$. At the end of a phase, Alg migrates the file to a point A_1 that minimizes the function

$$f(x) = [A_0, x] + \beta \cdot [x, \mathcal{R}] \ .$$

As in the amortized analysis of the algorithm Mtlm [8], we will use a potential function equal to ϕ times the distance between the files of Alg and Opt, where ϕ is a parameter used in the analysis. We let O_0 and O_1 denote the initial and final position of Opt during the studied phase, respectively. Then, the amortized cost of Alg in a single phase is $C_{\text{ALG}} = \delta \cdot [A_0, \mathcal{R}] + [A_0, A_1] + \phi([A_1, O_1] - [A_0, O_0])$.

The following factor-revealing LP mimics the proof given in [8]. Namely, it encodes inequalities that are true for any phase and a graph on which Alg can be run. Its goal is to maximize the ratio between C_{ALG} and C_{OPT}: as an instance can be scaled, we set $C_{\text{OPT}} = 1$ and we maximize C_{ALG}. Let $V = \{A_0, A_1, O_0, O_1\}$ and $V' = V \cup \{\mathcal{R}\}$.

maximize C_{ALG}

subject to:

$$C_{\text{ALG}} = \delta \cdot [A_0, \mathcal{R}] + [A_0, A_1] + \phi \cdot ([A_1, O_1] - [A_0, O_0])$$
$$C_{\text{OPT}} = 1$$
$$C_{\text{OPT}} = C^{\text{req}}_{\text{OPT}} + C^{\text{move}}_{\text{OPT}}$$
$$C^{\text{move}}_{\text{OPT}} \geq [O_0, O_1]$$
$$2 \cdot C^{\text{req}}_{\text{OPT}} + \delta \cdot C^{\text{move}}_{\text{OPT}} \geq \delta \cdot [O_0, \mathcal{R}] + \delta \cdot [O_1, \mathcal{R}]$$
$$f(A_1) \leq f(v) \qquad \text{for all } v \in V$$
$$0 \leq [v_1, v_3] \leq [v_1, v_2] + [v_2, v_3] \qquad \text{for all } v_1, v_2, v_3 \in V'$$

As $\mathcal{R}$ is a set of requests, it does not necessarily correspond to a single point in the studied metric. Nevertheless, our notion of average distances (i.e., $[v_1, v_2]$) allows us to write the triangle inequalities for any pair of objects from set $V \cup \{\mathcal{R}\}$.

In the LP above, $C^{\text{req}}_{\text{OPT}}$, $C^{\text{move}}_{\text{OPT}}$ denote the cost of Opt for serving the request and the cost of Opt for migrating the file, respectively. The inequality $2 \cdot C^{\text{req}}_{\text{OPT}} + \delta \cdot C^{\text{move}}_{\text{OPT}} \geq \delta \cdot [O_0, \mathcal{R}] + \delta \cdot [O_1, \mathcal{R}]$ is a counterpart of the relation guaranteed by Lemma 3.

For any choice of parameters β, δ, and ϕ, the LP above finds an instance that maximizes the competitive ratio of Alg. Note that such instance is not necessarily a certificate that Alg indeed performs poorly: in particular, inequalities that lower-bound the cost of Opt might not be tight. However, the opposite is true: if the value of C_{ALG} returned by the LP is ξ, then for any possible instance the ratio is at most ξ.

Let $c_0 = 1.841$ be the phase length of Mtlm. Setting $\delta = c_0$ and $\beta = \phi = 1 + c_0$ yields that the optimal value of the LP is $R_0 \approx 4.086$, which can be interpreted as a numerical counterpart of the original analysis in [8]. To obtain a formal mathematical proof, one may take a dual solution to the LP. It gives the coefficients that multiplied by the corresponding LP inequalities and summed over all inequalities yield a proof that the amortized cost of Mtlm in any phase is at most R_0 times the cost of Opt. Summed over all phases, this implies that Mtlm is R_0-competitive.

Among other advantages, this approach allows us to numerically find the instances that are tight for the current analysis (cf. Section 2 and Figure 1): linear and bipartite instances can be obtained this way.

4.2 LP analysis of DLM-like algorithms

Now we show how to adapt the LP from the previous section to analyze Dlm-type algorithms. Recall that after $1.75\,D$ requests, Dlm evaluates the geometry of the so-far-received requests and decides whether to continue this phase or not. Although the final parameters of Dlm are elegant numbers (multiplicities of 1/4), they were obtained by a tedious optimization process using the LP we present below. Furthermore, the LP below does not give us an explicit rule for continuing the phase; it only tells that Dlm is successful either in a short or in a long phase.

Recall that in a phase, Dlm considers three groups of consecutive $\delta_i \cdot D$ requests: $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$, where δ_1, δ_2, and δ_3 are parameters of Dlm. First, assume that Dlm always processes three parts and afterwards it moves the file to a point A_3 that minimizes the function

$$h(x) = [A_0, x] + \beta_1 \cdot [x, \mathcal{R}_1] + \beta_2 \cdot [x, \mathcal{R}_2] + \beta_3 \cdot [x, \mathcal{R}_3] \ ,$$

where β_i are parameters that we choose later. We denote the strategy of an optimal algorithm by OptL (short for Opt-Long). Let O_0^L, O_1^L, O_2^L and O_3^L denote the trajectory of OptL (O_0^L is the initial position of the OptL's file at the beginning of the phase, and O_i^L is its position right after the i-th part of the phase). Analogously to the previous section, we obtain the following LP.

maximize C_{ALGL}

subject to:

$$C_{\text{ALGL}} = [A_0, A_3] + \textstyle\sum_{i=1,2,3} \delta_i \cdot [A_0, \mathcal{R}_i] + \phi \cdot ([A_3, O_3^L] - [A_0, O_0^L])$$

$$C_{\text{OPTL}} = 1$$

$$C_{\text{OPTL}} = \textstyle\sum_{i=1,2,3} (C^{\text{req}}_{\text{OPTL}}(i) + C^{\text{move}}_{\text{OPTL}}(i))$$

$$C^{\text{move}}_{\text{OPTL}}(i) \geq [O_{i-1}^L, O_i^L] \qquad \text{for } i = 1, 2, 3$$

$$2 \cdot C^{\text{req}}_{\text{OPTL}}(i) + \delta \cdot C^{\text{move}}_{\text{OPTL}}(i) \geq \delta_i \cdot [O_{i-1}^L, \mathcal{R}_i] + \delta_i \cdot [O_i^L, \mathcal{R}_i] \qquad \text{for } i = 1, 2, 3$$

$$h(A_3) \leq h(v) \qquad \text{for all } v \in V$$

$$0 \leq [v_1, v_3] \leq [v_1, v_2] + [v_2, v_3] \qquad \text{for all } v_1, v_2, v_3 \in V'$$

This time $V = \{A_0, A_3, O_0^L, O_1^L, O_2^L, O_3^L\}$ and $V' = V \cup \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$.

We note that such parameterization alone does not improve the competitive ratio, i.e., for any choice of parameters δ_i and β_i, the objective value of the LP above is at least $R_0 \approx 4.086$.

However, as stated in Section 2.1, DLM verifies if after two parts it can migrate its file to a node A_2 being the minimizer of the function

$$g(x) = [A_0, x] + \beta'_1 \cdot [x, \mathcal{R}_1] + \beta'_2 \cdot [x, \mathcal{R}_2] \ ,$$

where β'_i are parameters that we choose later.

In our analysis, we gave an explicit rule whether the migration to A_2 should take place. However, for our LP-based approach, we follow a slightly different scheme. Namely, if the migration to A_2 guarantees that the amortized cost in the short phase (the first two parts) is at most 4 times the cost of *any* strategy for the short phase, then DLM may move to A_2 and we immediately achieve competitive ratio 4 on the short phase. Otherwise, we may add additional constraints to the LP, stating that the competitive ratio of an algorithm which moves to A_2 is at least 4 (against any chosen strategy OPTS). Analogously to OPTL, the trajectory of OPTS is described by three points: O_0^S, O_1^S, and O_2^S. This allows us to strengthen our LP by adding the following inequalities:

$$\begin{aligned}
&C_{\text{ALGS}} = [A_0, A_2] + \textstyle\sum_{i=1,2} \delta_i \cdot [A_0, \mathcal{R}_i] + \phi \cdot ([A_2, O_2^S] - [A_0, O_0^S]) \\
&C_{\text{OPTS}} = \textstyle\sum_{i=1,2} (C^{\text{req}}_{\text{OPTS}}(i) + C^{\text{move}}_{\text{OPTS}}(i)) \\
&C^{\text{move}}_{\text{OPTS}}(i) \geq [O_{i-1}^S, O_i^S] && \text{for } i = 1, 2 \\
&2 \cdot C^{\text{req}}_{\text{OPTS}}(i) + \delta \cdot C^{\text{move}}_{\text{OPTS}}(i) \geq \delta_i \cdot [O_{i-1}^S, \mathcal{R}_i] + \delta_i \cdot [O_i^S, \mathcal{R}_i] && \text{for } i = 1, 2 \\
&g(A_2) \leq g(v) && \text{for all } v \in V \\
&C_{\text{ALGS}} \geq 4 \cdot C_{\text{OPTS}}
\end{aligned}$$

We also change V to $\{A_0, A_3, O_0^L, O_1^L, O_2^L, O_3^L, O_0^S, O_1^S, O_2^S\}$, both in new and in old inequalities.

When we choose $\phi = 3$, fix phase length parameters to be $\delta_1 = 1$, $\delta_2 = 0.75$, $\delta_3 = 0.5$ and parameters for functions g and h to be $\beta'_1 = 2$, $\beta'_2 = 1$, $\beta_1 = 1, \beta_2 = 0.25$ and $\beta_3 = 0.75$, we obtain that the value of the above LP is 4. Again, this can be interpreted as a numerical argument that DLM is indeed a 4-competitive algorithm.

5 Conclusions

While in the last decade factor-revealing LPs became a standard tool for analysis of approximation algorithms, their application to online algorithms so far have been limited to online bipartite matching and its variants (see, e.g., [24, 20]) and for showing lower bounds [5]. In this paper, we successfully used the factor-revealing LP to bound the competitive ratio of an algorithm for an online problem defined on an arbitrary metric space. We believe that similar approaches could yield improvements also for other online graph problems.

References

1 Susanne Albers and Hisashi Koga. Page migration with limited local memory capacity. In *Proc. 4th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 147–158, 1995.

2 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Competitive distributed file allocation. In *Proc. 25th ACM Symp. on Theory of Computing (STOC)*, pages 164–173, 1993.

3 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Heat & Dump: Competitive distributed paging. In *Proc. 34th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 22–31, 1993.

4 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Distributed paging for general networks. *Journal of Algorithms*, 28(1):67–104, 1998.

5 Yossi Azar, Ilan Reuven Cohen, and Alan Roytman. Online lower bounds via duality. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1038–1050, 2017.

6 Yair Bartal. *Competitive Analysis of Distributed On-line Problems – Distributed Paging*. PhD thesis, Tel-Aviv University, 1995.

7 Yair Bartal. Distributed paging. In *Dagstul Workshop on On-line Algorithms*, pages 97–117, 1996.

8 Yair Bartal, Moses Charikar, and Piotr Indyk. On page migration and other relaxed task systems. *Theoretical Computer Science*, 268(1):43–66, 2001. Also appeared in *Proc. of the 8th SODA*, pages 43–52, 1997.

9 Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995.

10 Shai Ben-David, Allan Borodin, Richard M. Karp, Gabor Tardos, and Avi Wigderson. On the power of randomization in online algorithms. *Algorithmica*, 11(1):2–14, 1994.

11 Marcin Bienkowski. Migrating and replicating data in networks. *Computer Science – Research and Development*, 27(3):169–179, 2012.

12 Marcin Bienkowski, Jaroslaw Byrka, Miroslaw Korzeniowski, and Friedhelm Meyer auf der Heide. Optimal algorithms for page migration in dynamic networks. *Journal of Discrete Algorithms*, 7(4):545–569, 2009.

13 David L. Black and Daniel D. Sleator. Competitive algorithms for replication and migration problems. Technical Report CMU-CS-89-201, Department of Computer Science, Carnegie-Mellon University, 1989.

14 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

15 Marek Chrobak, Lawrence L. Larmore, Nick Reingold, and Jeffery Westbrook. Page migration algorithms using work functions. *Journal of Algorithms*, 24(1):124–157, 1997.

16 Bezalel Gavish and Olivia R. Liu Sheng. Dynamic file migration in distributed computer systems. *Communications of the ACM*, 33(2):177–189, 1990.

17 Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

18 Carsten Lund, Nick Reingold, Jeffery Westbrook, and Dicky C. K. Yan. Competitive on-line algorithms for distributed data management. *SIAM Journal on Computing*, 28(3):1086–1111, 1999.

19 Bruce M. Maggs, Friedhelm Meyer auf der Heide, Berthold Vöcking, and Matthias Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 284–293, 1997.

20 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, pages 597–606, 2011.

21 Akira Matsubayashi. Uniform page migration on general networks. *International Journal of Pure and Applied Mathematics*, 42(2):161–168, 2008.

22 Akira Matsubayashi. A 3+Omega(1) lower bound for page migration. In *Proc. 3rd Int. Symp. on Computing and Networking (CANDAR)*, pages 314–320, 2015.

23 Akira Matsubayashi. Asymptotically optimal online page migration on three points. *Algorithmica*, 71(4):1035–1064, 2015.

24 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5), 2007.

25 Friedhelm Meyer auf der Heide, Berthold Vöcking, and Matthias Westermann. Provably good and practical strategies for non-uniform data management in networks. In *Proc. 7th European Symp. on Algorithms (ESA)*, pages 89–100, 1999.

26 Jeffery Westbrook. Randomized algorithms for the multiprocessor page migration. *SIAM Journal on Computing*, 23:951–965, 1994.

The Infinite Server Problem*†

Christian Coester[1], Elias Koutsoupias[2], and Philip Lazos[3]

1 **Department of Computer Science, University of Oxford, Oxford, UK**
`christian.coester@cs.ox.ac.uk`
2 **Department of Computer Science, University of Oxford, Oxford, UK**
`elias.koutsoupias@cs.ox.ac.uk`
3 **Department of Computer Science, University of Oxford, Oxford, UK**
`filippos.lazos@cs.ox.ac.uk`

Abstract

We study a variant of the k-server problem, the infinite server problem, in which infinitely many servers reside initially at a particular point of the metric space and serve a sequence of requests. In the framework of competitive analysis, we show a surprisingly tight connection between this problem and the (h, k)-server problem, in which an online algorithm with k servers competes against an offline algorithm with h servers. Specifically, we show that the infinite server problem has bounded competitive ratio if and only if the (h, k)-server problem has bounded competitive ratio for some $k = O(h)$. We give a lower bound of 3.146 for the competitive ratio of the infinite server problem, which implies the same lower bound for the (h, k)-server problem even when $k/h \to \infty$ and holds also for the line metric; the previous known bounds were 2.4 for general metric spaces and 2 for the line. For weighted trees and layered graphs we obtain upper bounds, although they depend on the depth. Of particular interest is the infinite server problem on the line, which we show to be equivalent to the seemingly easier case in which all requests are in a fixed bounded interval away from the original position of the servers. This is a special case of a more general reduction from arbitrary metric spaces to bounded subspaces. Unfortunately, classical approaches (double coverage and generalizations, work function algorithm, balancing algorithms) fail even for this special case.

1998 ACM Subject Classification F.1.2 Modes of Computation (Online Computation)

Keywords and phrases Online Algorithms, k-Server, Resource Augmentation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.14

1 Introduction

The k-server problem is a fundamental well-studied online problem [19, 16]. In this problem k servers serve a sequence of requests. The servers reside at k points of a metric space M and requests are simply points of M. Serving a request entails moving one of the servers to the request. The objective is to minimize the total distance traveled by the servers. The most interesting variant of the problem is its online version, in which the requests appear one-by-one and the online algorithm must decide how to serve a request without knowing the future requests. It is known that the deterministic k-server problem has competitive ratio between k and $2k - 1$ for every metric space with at least $k + 1$ distinct points [19, 17].

In this paper, we study the *infinite server problem*, the variant of the k-server problem in which there are infinitely many servers, all of them initially residing at a given point, the

* Full version available at `https://arxiv.org/abs/1702.08474`.
† Supported by the ERC Advanced Grant 321171 (ALGAME) and by EPSRC.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 14; pp. 14:1–14:14


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

source[1]. At first glance it may appear that the lower bound of k for the k-server problem would imply an unbounded competitive ratio for the infinite server problem. But consider, for example, the version of the k-server problem on uniform metric spaces (i. e. the distance between any two points is 1), and observe that the infinite server problem has competitive ratio 1 for this case.

The infinite server problem is closely related to the (h, k)-server problem, the resource augmentation version of the k-server problem in which the online algorithm has k servers and competes against an offline algorithm for $h \leq k$ servers. This model is also known as *weak adversaries* [2, 15]. One major open problem in competitive analysis is whether the (h, k)-server problem has bounded competitive ratio when $k \gg h$. Here we show a, perhaps surprising, tight connection between the infinite server problem and the (h, k)-server problem, which also allows us to improve lower bounds for the latter.

The infinite server problem is also a considerable generalization of the ski-rental problem, since the ski-rental problem is essentially a special case of the infinite server problem when the metric space is an isosceles triangle.

Besides its theoretical appeal, our three main reasons for investigating the infinite server problem are the following. First, the competitive ratio of the k-server problem goes to infinity as $k \to \infty$, but for $k = \infty$ it goes back to a small constant at least on some metric spaces. This suggests that the high competitive ratio of the k-server problem is somewhat artificial. Second, the problem allows to model applications where the number of servers is so high that it is not a limitation in practice, or where more servers can be bought. A price for buying new servers can be modeled easily by appropriate placement of the source in the metric space. Third, the relationship between the infinite server problem and the (h, k)-server problem allows for new ways to tackle the latter.

1.1 Previous Work

The k-server problem was first formulated by Manasse et al. [19], to generalize a variety of online settings whose stepwise cost had a 'metric'-like structure. They built on previous work by Sleator and Tarjan [20], the genesis of competitive analysis, on the paging problem. This problem can be easily recast as a k-server instance for the uniform metric and was already known to be k-competitive.

Manasse et al. [19] also showed that the competitive ratio of the k-server problem is at least k on any metric space with more than k points. They then proposed the renowned k-server conjecture, stating that this bound is tight. This has been shown to be true for $k = 2$ [19] and for several special metric spaces [6, 7, 18, 19, 20]. A stream of refinements [12, 3] led to better competitive ratios for general metric spaces until [17] showed that a competitive ratio of $2k - 1$ can be achieved on any metric space. Chasing the competitive ratio for the deterministic (and randomized) k-server problem has been pivotal for the development of competitive analysis. For a more in depth view on the history of the k-server problem and further related work, we refer to [16].

In the weak adversaries setting, significantly less is known. For the (h, k)-server problem, the exact competitive ratio is $\frac{k}{k-h+1}$ on uniform metrics (equivalent to paging) [20] and weighted stars (equivalent to weighted paging) [21]. Bansal et al. [1] showed recently for weighted trees that the competitive ratio as $k/h \to \infty$ is bounded by a constant depending on the depth of the tree. On general metrics, the (h, k)-server problem is still poorly understood.

[1] We first learned about this problem from Kamal Jain [14].

No algorithm is known for general metrics that performs better than disabling the $k-h$ extra servers and using h servers only. In fact, for the line it was shown [2, 1] that the Double Coverage Algorithm and the Work Function Algorithm – despite achieving the optimal competitive ratio of h if $k=h$ [6, 4] – perform strictly worse in the resource augmentation setting than disabling the $k-h$ extra servers and applying the same algorithm to h servers only. For the case that h is not fixed, the Work Function Algorithm was shown to be $2h$-competitive simultaneously against any number $h \leq k$ of offline servers [15].

In terms of lower bounds, it is known that unlike for (weighted) paging, the competitive ratio does not converge to 1 on general metrics even as $k/h \to \infty$. Prior to this work, the best known lower bounds were 2 on the line, due to Bar-Noy and Schieber (see [5, p. 175]), and 2.4 for general metrics as shown recently by Bansal et al. [1].

The closest publication to this work is by Csirik et al. [10], which studies a problem that is essentially the special case of the infinite server problem on the uniform metric space augmented by a far away source. It is cast as a paging problem where new cache slots can be bought at a fixed price per unit and gives matching upper and lower bounds of ≈ 3.146 on the competitive ratio.

1.2 Our Results

Our main result is an equivalence theorem between the infinite server problem and the (h,k)-server problem, presented in Section 2. It states that the infinite server problem is competitive on every metric space if and only if the (h,k)-server problem is $O(1)$-competitive on every metric space as $k/h \to \infty$. We show further that it is not even necessary to let k/h tend to infinity because in the positive case, there must also exist some $k = O(h)$. The theorem holds also if "every metric space" is replaced by "the real line".

In Section 3 we present upper and lower bounds on the competitive ratio of the infinite server problem on a variety of metric spaces. Extending the work in [10], we present a tight lower bound for non-discrete spaces, which is then turned into a 3.146 lower bound for the (h,k) setting. To our knowledge, this is the largest bound on the weak adversaries setting for any metric space, as $k/h \to \infty$. We show how recent work by Bansal et al. [1] can be adapted to give an upper bound on the competitive ratio of the infinite server problem on bounded-depth weighted trees. We also consider layered graph metrics, which are equivalent (up to a factor of 2) to general graph metrics. We have not settled the case for their competitive ratio, but we present a natural algorithm with tight analysis and pose challenges for further research. The main open problem is whether there exists a metric space on which the infinite server problem is not competitive.

In Section 4 we show how a variety of known algorithms such as the work function and balancing algorithms fail for the infinite server problem, even on the real line. We focus in particular on a class of speed-adjusted variants of the well-known double coverage algorithm.

Finally, we present a useful reduction from arbitrary metric spaces to bounded subspaces in Section 5. In particular, the infinite server problem on the line is competitive if and only if it is competitive for the special case where requests are restricted to some bounded interval further away from the source.

1.3 Preliminaries

Let $M=(M,d)$ be a metric space and let s be a point of M. In the *infinite server problem on* (M,s), an unbounded number of servers starts at point s and serves a finite sequence

$\sigma = (\sigma_0 = s, \sigma_1, \sigma_2, \ldots, \sigma_m)$ of requests $\sigma_i \in M$. Serving a request entails moving one of the servers to it. The goal is to minimize the total distance traveled by the servers.

We drop s in the notation if the location of the source is not relevant or understood. We refer to the action of moving a server from the source to another point as *spawning*. Throughout this work we use the letter d for the metric associated with the metric space.

In the online setting, the requests are revealed one by one and need to be served immediately without knowledge of future requests. All algorithms considered in this paper are deterministic. An algorithm is called *lazy* if it moves only one server to serve a request at an unoccupied point and moves no server if the requested point is already covered. An algorithm is called *local* [9] if it moves a server from a to b only if there is no server at some other point c on a shortest path from a to b, i. e. with $d(a, b) = d(a, c) + d(c, b)$. It is easy to see that any algorithm can be turned into a lazy and local algorithm without increasing its cost (i. e. the total distance traveled by all servers).

For an algorithm ALG, we denote by $ALG(\sigma)$ its cost on the request sequence σ. Similarly, we write $OPT(\sigma)$ for the optimal (offline) cost.

An online algorithm ALG is *ρ-competitive* for $\rho \geq 1$ if $ALG(\sigma) \leq \rho OPT(\sigma) + c$ for all σ, where c is a constant independent of σ. The *competitive ratio* of an algorithm is the infimum of all such ρ. We say that an algorithm is *competitive* if it is ρ-competitive for some ρ. We also call an online problem itself (ρ-)competitive if it admits such an algorithm. If the additive term c in the definition is 0, then the algorithm is also called *strictly ρ-competitive* [11].

The *(h, k)-server problem on M* is defined like the infinite server problem except that the number of servers is k for the online algorithm and h for the optimal (offline) algorithm against whom it is compared in the definition of competitiveness. For this problem, the servers are not required to start at the same point, although a different initial configuration would only affect the additive term c. The problem is interesting only when $k \geq h$. The case $h = k$ is the standard k-server problem and the case $k \geq h$ is known as the *weak adversaries* model. One major open problem is to determine the competitive ratio of the (h, k)-server problem as k tends to infinity.

We will sometimes write OPT_h and OPT_∞ for the optimal offline algorithm, where the index specifies the number of servers available.

The following two propositions will be useful later in the paper.

▶ **Proposition 1.** *If for every metric space there exists a competitive algorithm for the infinite server problem, then there exists a universal competitive ratio ρ such that the infinite server problem is* strictly *ρ-competitive on every metric space.*

Proof. We first show the existence of ρ such that the infinite server problem is ρ-competitive (strictly or not) on every metric space. Suppose such ρ does not exist, then for every $n \in \mathbb{N}$ we can find a metric space M_n containing some point s_n such that the infinite server problem on (M_n, s_n) is not n-competitive. Consider the metric space obtained by taking the disjoint union of all spaces M_n and gluing all the points s_n together. The infinite server problem would not be competitive on this metric space, in contradiction to the assumption.

Analogously we can also find a universal constant c that works for all metric spaces as additive constant in the definition of ρ-competitiveness. A scaling argument shows that also $c = 0$ works. ◀

With a very similar argument we get:

▶ **Proposition 2.** *Let $k = k(h)$ be a function of h. Suppose that for every metric space M and for all h there exists an $O(1)$-competitive algorithm for the (h, k)-server problem on M. Then there exists a universal competitive ratio ρ such that the (h, k)-server problem is strictly ρ-competitive on every metric space if all servers start at the same point.*

2 Equivalence of Infinite Servers and Weak Adversaries

The main result of this section is the following tight connection between the infinite server problem and the weak adversaries model.

▶ **Theorem 3.** *The following are equivalent:*
1. *The infinite server problem is competitive.*
2. *The (h, k)-server problem is $O(1)$-competitive as $k/h \to \infty$.*
3. *For each h there exists $k = O(h)$ so that the (h, k)-server problem is $O(1)$-competitive.*

The three statements above are also equivalent if we fix the metric space to be the real line.

The implication "3 $\Longrightarrow$ 2" is trivial. The proof of the equivalence theorem consists in its core of two reductions. Theorem 4 contains the easier of the two reductions, which is from the infinite server problem to the k-server problem against weak adversaries ("2 $\Longrightarrow$ 1"). By Propositions 1 and 2, it suffices to consider only strictly competitive algorithms. Theorem 5 proves essentially the inverse for general metric spaces, and Theorem 8 specializes it to the line ("1 $\Longrightarrow$ 3").

As a corollary of the theorem we get the non-trivial implication "2 $\Longrightarrow$ 3", a potentially useful statement towards resolving the major open problem about weak adversaries: "Is Statement 2 true?" This highlights the importance of the infinite server problem.

▶ **Theorem 4.** *Fix a metric space M and consider algorithms with all servers starting at some $s \in M$. If for every h there exists $k = k(h)$ such that the (h, k)-server problem on M is strictly ρ-competitive, for some constant ρ, then there exists a strictly ρ-competitive online strategy for the infinite server problem on M.*

Proof. Let $ALG_{k(h)}$ denote an online algorithm with $k(h)$ servers that is strictly ρ-competitive against an optimal algorithm OPT_h for h servers, i. e.

$$ALG_{k(h)}(\sigma) \leq \rho OPT_h(\sigma) \tag{1}$$

for every request sequence σ. Without loss of generality, algorithm $ALG_{k(h)}$ is lazy.

For every request sequence σ, consider the equivalence relation $\equiv_\sigma$ on natural numbers in which $h \equiv_\sigma h'$ if and only if $ALG_{k(h)}(\sigma)$ and $ALG_{k(h')}(\sigma)$ serve σ in exactly the same way (i. e. , make exactly the same moves). To every σ, we associate an equivalence class $H(\sigma)$ of $\equiv_\sigma$ that satisfies
- $H(\sigma)$ is infinite,
- $H(\sigma r) \subseteq H(\sigma)$, for every request r.

This is done inductively in the length of σ (in a manner reminiscent of König's lemma) as follows: For the base case when σ is the empty request sequence, $H(\sigma) = \mathbb{N}$. For the induction step, suppose that we have defined $H(\sigma)$. Consider the equivalence classes of $\equiv_{\sigma r}$, a refinement of the equivalence classes of $\equiv_\sigma$. Since there are only finitely many possible ways to serve r, they partition $H(\sigma)$ into finitely many parts. At least one of these parts is infinite and we select it to be $H(\sigma r)$; if there is more than one such sets, we select one arbitrarily, say the lexicographically first.

Given such a mapping H, we define the online algorithm ALG_∞ which serves every σ in the same way as all the online algorithms $ALG_{k(h)}$ for $h \in H(\sigma)$. The second property of H guarantees that ALG_∞ is a well-defined online algorithm.

By construction, $ALG_\infty(\sigma) = ALG_{k(h)}(\sigma)$ for every $h \in H(\sigma)$. To finish the proof, observe that since $H(\sigma)$ is infinite, it contains some h greater than the length of σ, and for such an h we have $OPT_\infty(\sigma) = OPT_h(\sigma)$. Substituting these to (1), we see that ALG_∞ is strictly ρ-competitive. ◀

We now show the reduction from the k-server problem against weak adversaries to the infinite server problem on general metric spaces.

▶ **Theorem 5.** *If the infinite server problem on general metric spaces is strictly $\tilde{\rho}$-competitive, then there exists a constant ρ such that the (h,k)-server problem is ρ-competitive, for $k = O(h)$. In particular, for every $\epsilon > 0$, we can take $\rho = (3+\epsilon)\tilde{\rho}$ and any $k \geq (1+1/\epsilon)\tilde{\rho}h$.*

Proof. Fix some metric space M and a point $s \in M$. We will describe a strictly ρ-competitive algorithm for the (h,k)-server problem on M for the case that all servers start at s. This implies a (not necessarily strictly) ρ-competitive algorithm for any initial configuration.

The idea is to simulate a strictly $\tilde{\rho}$-competitive infinite server algorithm, but whenever it would spawn a $(k+1)$-st server, we bring all servers back to the origin and restart the algorithm. The problem is that the overhead cost for returning the servers to the origin, may be very high. To compensate for this, we assume that every time the servers return to the origin, they pretend to start from a different point further away from the origin. This motivates the following notation:

▶ **Definition 6.** Given a metric M, a point $s \in M$, and a value $w \geq 0$, we will use the notation $M_{s \oplus w}$ to denote the metric derived from M when we increase the distance of s from every other point by w; we will also denote the relocated point by $s \oplus w$.

Let ALG_∞ denote a strictly $\tilde{\rho}$-competitive online algorithm for the infinite server problem. We now define an online algorithm ALG_k for k servers (all starting at s). We will make use of the notation $A(\sigma; s)$ to denote the cost of algorithm A to serve the request sequence σ when all servers start at s.

▶ **Definition 7** (ALG_k derived from ALG_∞)**.** Algorithm ALG_k runs in phases with the initial phase being the 0th phase. At the beginning of every phase, all servers of ALG_k are at s. In every phase i, the algorithm simulates the infinite server algorithm ALG_∞, whose servers start at $s \oplus w_i$ for some $w_i \geq 0$. The parameters w_i are determined online, and initially $w_0 = 0$. Whenever ALG_∞ spawns a server from $s \oplus w_i$, algorithm ALG_k spawns a server from s.

The phase ends just before ALG_∞ spawns its $(k+1)$-st server or when the request sequence ends. In the former case, all servers of ALG_k return to s to start the $(i+1)$-st phase. To determine the starting point of the simulated algorithm of the next phase, we set

$$w_{i+1} = \epsilon \, \frac{OPT_h(\sigma_i; s)}{h}\,, \tag{2}$$

where σ_i is the sequence of requests during phase i.

Let n be the number of phases. The cost of ALG_k for the requests in phase $i < n$ is $ALG_\infty(\sigma_i; s \oplus w_i) - kw_i$; the last term is subtracted because the k servers do not have to actually travel the distance between $s \oplus w_i$ and s. However for the last phase no such term can be subtracted since we do not know how many servers are spawned during the phase, and we can only bound the cost from above by $ALG_\infty(\sigma_n; s \oplus w_n)$. The cost of returning the servers to s at the end of a phase can at most double the cost during the phase.

From this, we see that the total cost of ALG_k in phase i is

$$cost_i \leq \begin{cases} 2\left(ALG_\infty(\sigma_i; s \oplus w_i) - kw_i\right) & \text{for } i < n \\ ALG_\infty(\sigma_n; s \oplus w_n) & \text{for } i = n\,. \end{cases}$$

Since ALG_∞ is strictly $\tilde{\rho}$-competitive, we have

$$\begin{aligned} ALG_\infty(\sigma_i; s \oplus w_i) &\leq \tilde{\rho}\, OPT_\infty(\sigma_i; s \oplus w_i) \\ &\leq \tilde{\rho}\, OPT_h(\sigma_i; s \oplus w_i) \\ &\leq \tilde{\rho}\, (OPT_h(\sigma_i; s) + hw_i) \end{aligned}$$

and substituting this in the expression for the cost, we can bound the total cost by

$$\begin{aligned} ALG_k(\sigma; s) = \sum_{i=0}^{n} cost_i &\leq 2 \sum_{i=0}^{n-1} (\tilde{\rho}(OPT_h(\sigma_i; s) + hw_i) - kw_i) + \tilde{\rho}(OPT_h(\sigma_n; s) + hw_n) \\ &= 2 \sum_{i=0}^{n-1} (\tilde{\rho} OPT_h(\sigma_i; s) - (k - \tilde{\rho}h)w_i) + \tilde{\rho} OPT_h(\sigma_n; s) + \tilde{\rho} h w_n \,. \end{aligned}$$

The parameters w_i and k were selected so that the summation telescopes, and we are left with

$$\begin{aligned} ALG_k(\sigma; s) &\leq 2\,\tilde{\rho}\, OPT_h(\sigma_{n-1}; s) + \tilde{\rho}\, OPT_h(\sigma_n; s) + \tilde{\rho}\,\epsilon\, OPT_h(\sigma_{n-1}; s) \\ &\leq (3 + \epsilon)\,\tilde{\rho}\, OPT_h(\sigma; s)\,. \end{aligned}$$

◀

The previous reduction requires the infinite server problem to be competitive on *every* metric space. The following variant only requires the infinite server problem to be competitive on the line.

▶ **Theorem 8.** *If the infinite server problem on the line is ρ-competitive, then for every $h \in \mathbb{N}$ and $\epsilon > 0$, the (h, k)-server problem on the line is $(3 + \epsilon)\rho$-competitive, when $k \geq 2\lceil(1 + 1/\epsilon)\rho h\rceil$.*

Proof. A straightforward adaptation of the proof of the previous lemma, shows the existence of a $(3 + \epsilon)\rho$-competitive algorithm for the interval $[0, \infty)$, when $k \geq 2(1 + 1/\epsilon)\rho h$. By doubling the number of online servers so that half of them are used in each half-line, we get a $(3 + \epsilon)\rho$-competitive algorithm for the entire line, when $k \geq 2\lceil(1 + 1/\epsilon)\rho h\rceil$.

Note that the proof assumes strictly competitive algorithms. But, by a straightforward scaling argument, if the infinite server problem on the line is ρ-competitive, then it is also strictly ρ-competitive. This in turn implies a strictly ρ-competitive online algorithm for $M_{0 \oplus w}$, since this space is isometric to the subspace $\{-w\} \cup (0, \infty)$ of the line. ◀

In the next section we look at some particular metric spaces and give upper and lower bounds on the competitive ratio.

3 Upper and Lower Bounds

Unlike the k-server problem, which is 1-competitive if and only if the metric spaces has at most k points and conjectured k-competitive otherwise, the situation is more diverse for the infinite server problem. For example, on uniform metric spaces (where all distances are the same) the problem is trivially 1-competitive even if the metric space consists of uncountably many points. This is because an optimal strategy in this case is to spawn a server to every requested point. More generally, this strategy achieves a finite competitive ratio on any metric space where distances are bounded from below and above by positive constants. This suggests that statements about the competitive ratio for the infinite server problem cannot be as simple as the (conjectured) dichotomy for the k-server problem, which depends only on the number of points of the metric space. In this section we derive bounds on the competitive ratio for particular classes of metric spaces.

3.1 Weighted Trees

We consider the infinite server problem on metric spaces that can be modeled by edge-weighted trees. The points of the metric space are the nodes of the tree, and the distance between two nodes is the sum of edge weights along their connecting path. We choose the source of the metric space as the root of the tree, and define the depth of the tree as the maximal number of edges from the root to a leaf. The number of nodes can be infinite (otherwise the infinite server problem is trivially 1-competitive), but we assume the depth to be finite.

An upper bound on the competitive ratio of such trees follows easily from an upper bound for the (h, k)-server on such trees [1] and the equivalence theorem:

▶ **Theorem 9.** *The competitive ratio of the infinite server problem on trees of depth d is at most $O(2^d \cdot d)$.*

Proof. Bansal et al. [1, Theorem 1.3] showed that the competitive ratio of the (h, k)-server problem on trees of depth d is at most $O(2^d \cdot d)$ provided that k/h is large enough. Inspection of the proof in [1] shows that if all servers start at the root, it is in fact strictly $O(2^d \cdot d)$-competitive. Thus, Theorem 4 implies the result for the infinite server problem. ◀

3.2 Non-Discrete Spaces and Spaces with Small Infinite Subspaces

The following theorem gives a lower bound of 3.146 on the competitive ratio of the infinite server problem on any metric space containing an infinite subspace of a diameter that is small compared to the subspace's distance from the source. For example, every non-discrete metric space has this property (unless the source is the only non-discrete point), since non-discrete metric spaces contain infinite subspaces of arbitrarily small diameter. The theorem is a generalization of such a lower bound established in [10] for a variant of the paging problem where cache cells can be bought. Crucial parts of the subsequent proof are as in [10].

▶ **Theorem 10.** *Let M be a metric space containing an infinite subspace $M_0 \subset M$ of finite diameter δ and a point $s \in M \setminus M_0$ such that the infimum Δ of distances between s and points in M_0 is positive. Let $\lambda > 3.146$ be the largest real solution to*

$$\lambda = 2 + \ln \lambda \,. \tag{3}$$

The competitive ratio of any deterministic online algorithm for the infinite server problem on (M, s) is bounded from below by a value that converges to λ as $\Delta/\delta \to \infty$. In particular, the competitive ratio is at least λ if $M \setminus \{s\}$ contains a non-discrete part.

Proof. By scaling the metric, we can assume that $\delta = 1$. Let $p_1, p_2, p_3, \ldots$ be infinitely many distinct points in M_0.

Fix some lazy deterministic online algorithm ALG. We consider the request sequence that always requests the point p_i with i minimal such that p_i is not occupied by a server of ALG. We call a move of a server between two points in M_0 *local* (i. e. every move that does not spawn is local). Let f_j be the cumulative cost of local moves incurred to ALG until it spawns its jth server. Let σ_k be this request sequence that is stopped right after ALG spawns its kth server, for some large k. The total online cost is

$$ALG(\sigma_k) \geq k\Delta + f_k \,. \tag{4}$$

Let $h = \lceil k/\lambda \rceil$. We consider several offline algorithms that start behaving the same way, so we think of it as one algorithm initially that is forked into several algorithms later. The

offline algorithms make use of only h servers and they begin by spawning them to the points $p_1, \dots, p_h$. They do not need to move any servers until ALG spawns its hth server. Whenever ALG spawns its jth server for some $j \geq h$, every offline algorithm is forked to h distinct algorithms: Each of them moves a different server to p_{j+1} (to prepare for the next request, which will be at p_{j+1}). We will keep the invariant that each offline algorithm already has a server at the next request. To this end, whenever ALG does a local move from p to p', every offline algorithm that does not have a server at p moves a server from p' to p; note that the algorithm had a server at p' by the invariant, and the next request will be at p.

When ALG has j spawned servers ($j \geq h$), the offline algorithms are in $\binom{j}{h-1}$ different configurations, each of which occurs equally often among them. If ALG does a local move from p to p', there are $\binom{j-1}{h-1}$ different offline configurations for which a local move is made in the opposite direction. Thus, for each local move by ALG while having j servers in total, a portion $\binom{j-1}{h-1}/\binom{j}{h-1} = \frac{j-h+1}{j}$ of the offline algorithms move a server in the opposite direction for the same cost.

We use the average cost of all offline algorithms we considered as an upper bound on the optimal cost. The cost of spawning h servers is at most $h(\Delta+1)$, and the average cost while ALG has j spawned servers (for $j = h, \dots, k-1$) is at most $\frac{j-h+1}{j}(f_{j+1} - f_j) + 1$ (with the "+1" coming from the move when offline algorithms fork). Hence,

$$\begin{aligned} OPT(\sigma_k) &\leq h(\Delta+1) + k - h + \sum_{j=h}^{k-1} \frac{j-h+1}{j}(f_{j+1} - f_j)\,, \\ &\leq h\Delta + k + \frac{k-h}{k-1} f_k - \frac{f_h}{h} - \sum_{j=h+1}^{k-1} \frac{h-1}{j(j-1)} f_j\,, \end{aligned}$$

Note that $\frac{f_k}{k}$ is bounded from above because otherwise ALG would not be competitive, and it is bounded from below by 0. Thus, $L = \liminf_{k\to\infty} \frac{f_k}{k}$ exists. In the following we use the asymptotic notation $o(1)$ for terms that disappear as $k \to \infty$. We can choose arbitrarily large values of k such that $\frac{f_k}{k} = L + o(1)$. Since $h = \lceil k/\lambda \rceil$, we have $\frac{f_j}{j} \geq L + o(1)$ for all $j \geq h$. Moreover, $\sum_{j=h+1}^{k-1} \frac{1}{j-1} = \ln(\lambda) + o(1)$. This allows us to simplify the previous bound to

$$\begin{aligned} OPT(\sigma_k) &\leq \frac{k}{\lambda}\Big(\Delta + \lambda + \big(\lambda - 1 - \ln(\lambda)\big)L + o(1)\Big) \\ &= \frac{k}{\lambda}\big(\Delta + L + \lambda + o(1)\big)\,, \end{aligned}$$

where the last step uses equation (3).

The competitive ratio is at least

$$\begin{aligned} \frac{ALG(\sigma_k) + O(1)}{OPT(\sigma_k)} &\geq \frac{k\Delta + f_k + O(1)}{\frac{k}{\lambda}\big(\Delta + L + \lambda + o(1)\big)} \\ &= \lambda \cdot \frac{\Delta + L}{\Delta + L + \lambda} + o(1)\,. \end{aligned}$$

The fraction in the last term tends to 1 as $\Delta \to \infty$. ◀

This bound is tight due to a matching upper bound in [10] that shows (translated to the terminology of the infinite server problem) that a competitive ratio of λ can be achieved on metric spaces where all pairwise distances are 1 except that the source is at some larger distance Δ from the other points.

The previous theorem together with the equivalence theorem also allows us to obtain a new lower bound for the k-server problem against weak adversaries.

▶ **Corollary 11.** *For sufficiently large h, there is no* 3.146*-competitive algorithm for the (h, k)-server problem on the line, even if $k \to \infty$.*

Proof. By a scaling argument it is easy to see that if the infinite server problem on the line is ρ-competitive, then it is also *strictly* ρ-competitive. Thus, the statement follows from Theorems 4 and 10. ◀

This improves upon both the previous best known lower bounds of 2 for this problem on the line [5, p. 175] and 2.4 on general metric spaces [1].

3.3 Layered Graphs

A *layered graph of depth D* is a graph whose (potentially infinitely many) nodes can be arranged in layers $0, 1, \ldots, D$ so that all edges run between adjacent layers and each node – except for a single node in layer 0 – is connected to at least one node of the previous layer. The induced metric space is the set of nodes with the distance being the minimal number of edges of a connecting path. For the purposes of the infinite server problem, the single node in layer 0 is the source. We assume $D \geq 2$ to avoid trivial cases.

Note that a connected graph is layered if and only if it is bipartite. Moreover, any graph can be embedded into a bipartite graph by adding a new node in the middle of each edge. So essentially, layered graphs capture *all* graph metrics.

Let *Move Only Outwards (MOO)* be some lazy and local algorithm for the infinite server problem on layered graphs that moves servers along edges only in the direction away from the source. Not surprisingly, the competitive ratio of this simple algorithm is quite bad and we show that it is exactly $D - 1/2$. Nonetheless, at least for $D \leq 3$ this is actually the optimal competitive ratio.

▶ **Theorem 12.** *The competitive ratio of MOO is exactly $D - \frac{1}{2}$.*

▶ **Theorem 13.** *The competitive ratio of the infinite server problem on layered graphs of depth D is exactly* 1.5 *for $D = 2$, exactly* 2.5 *for $D = 3$ and at least* 3 *for $D \geq 4$.*

Both theorems are proved in the full version of this paper. It remains an open problem to close the gap between the lower bound of 3 and the upper bound of 3.5 for $D = 4$. More importantly, we are interested in the question whether an algorithm better than MOO exists for large D, achieving a competitive ratio of less than $D - 1/2$ on any layered graph of depth D. Note that if no algorithm with a competitive ratio of $O(1)$ as $D \to \infty$ exists, then the infinite server problem on general metric spaces would not be competitive.

For large D, the lower bound of 3 is certainly not tight: Consider a layered graph where each layer contains one node except that the bottom layer contains infinitely many nodes. By Theorem 10 (and a matching upper bound shown in [10]), the competitive ratio on this graph converges to $\lambda \approx 3.146$ as $D \to \infty$.

4 Algorithms with Unbounded Competitive Ratio

We examine the performance of classical algorithms known for the k-server problem when applied to the infinite server problem, focusing on the line as a particularly appealing metric space. We also consider a generalization of the Double Coverage algorithm for the line with adjusted server speeds. This idea has proved successful for the (h, k)-server problem (and hence the infinite server problem) on weighted trees [1]. However, neither of these algorithms is competitive for the infinite server problem on the line. Proofs of the results of this section

as well as the definitions of the algorithms of the following theorem can be found in the full version of this paper.

▶ **Theorem 14.** *The Work Function Algorithm [8, 17], Balance [19] and Balance2 [13] are not competitive for the infinite server problem on the line.*

Perhaps more surprising than for the above three algorithms is that a class of algorithms extending the Double Coverage (DC) algorithm [6] is also not competitive for the infinite server problem. The basic DC algorithm on the line serves each request by an adjacent server. If the request lies between two servers, both servers move towards it at equal speed until one of them reaches the request. A sensible extension of this algorithm seems to be to give different speeds to servers, so that they move away from the source faster than towards it.

We consider here only the half-line $[0, \infty)$ with the source at the left border 0. Let x_i be the position of the ith server from the right. We use the notation x_i both for its position and for the server itself. As servers do not overtake each other, x_i is the ith spawned server. Let $\mathcal{S} = \{s_i \geq 1 \mid i \in \mathbb{N} \text{ and } i \geq 2\}$ for a monotonic (non-decreasing or non-increasing) sequence of speeds s_i. The algorithm $\mathcal{S}$-DC is defined as follows:

- If there exist servers x_{i+1} and x_i to the left and right of the request, move them towards it with speeds s_{i+1} and 1 respectively until one of the two reaches it.
- If a request does not have a server to its right, move the rightmost server to the request.

If $s_i = 1$ for all i, this is precisely the original DC algorithm.

▶ **Theorem 15.** *Algorithm $\mathcal{S}$-DC is not competitive for any $\mathcal{S}$.*

The intuitive reason is that servers move to the right either too slowly or too fast: Imagine repeatedly requesting the same n points in some small interval away from the source, until $\mathcal{S}$-DC covers all n points. One case is that $\mathcal{S}$-DC spawns too slowly and is therefore defeated by an adversary covering these n positions immediately with n servers. In the other case, the adversary will also use n servers to cover the initial group of requests and then shift its group of servers slowly towards the source, always making requests at the new positions of these offline servers. As $\mathcal{S}$-DC tries to cover the new requests, it is tricked into spawning too many servers. Both cases lead to an unbounded competitive ratio.

5 Reduction to Bounded Spaces

In this section we show a reduction from the infinite server problem on general metric spaces to bounded subspaces. Specifically, a metric space can be partitioned into "rings" of points whose distance from the source is between r^n and r^{n+1}, where $r > 1$ is fixed and $n \in \mathbb{Z}$. We show that if the infinite server problem is strictly ρ-competitive on each ring, then it is competitive on the entire metric space.

▶ **Theorem 16.** *Let M be a metric space and $s \in M$ and let $r > 1$. For $n \in \mathbb{Z}$ let $M_n = \{s\} \cup \{p \in M \mid d(s,p) \in [r^n, r^{n+1})\}$. If for each n the infinite server problem on (M_n, s) is strictly ρ-competitive, then on (M, s) it is strictly $\frac{4r-1}{r-1}\rho$-competitive.*

Proof. Let ALG_n be a ρ-competitive algorithm for the infinite server problem on (M_n, s).

For a request sequence σ, let σ_n be the subsequence of requests in M_n. Let ALG be the algorithm for (M, s) that uses different servers for each of the subsequences σ_n and serves them independently according to ALG_n.

The total online cost is $ALG(\sigma) = \sum_n ALG_n(\sigma_n) \le \rho \sum_n OPT(\sigma_n)$. To finish the proof, it suffices to show that

$$\sum_n OPT(\sigma_n) \le \frac{4r-1}{r-1} OPT(\sigma)\,. \tag{5}$$

Thus, we only need to analyze the offline cost. We do this for each offline server separately. Fix some offline server x. Let N_0 and N_1 be the minimal and maximal values of n such that x visits M_n. We can assume without loss of generality (by adding virtual points to the metric space) that whenever x moves from M_n to $M_{n'}$ for some $n < n'$, it travels across points $p_{n+1}, p_{n+2}, \ldots, p_{n'}$ with $d(s, p_i) = r^i$, and similarly for $n > n'$.

The movements of server x can be tracked by many servers, one server x_n in every set M_n for $N_0 \le n \le N_1$. When server x is in M_n, server x_n is exactly at the same position tracking the movement of x. When server x exits M_n at some point p at the boundary to M_{n-1} or M_{n+1}, server x_n freezes at p. The movement cost of x_n can be partitioned into the cost of deploying x_n at the first point visited in M_n, the tracking cost within M_n, and the cost of of relocating x_n whenever x re-enters M_n at a location different from the last exiting location.

The total tracking cost of all servers x_n is bounded by the distance traveled by x. The cost of deploying all servers x_n is $\sum_{n=N_0}^{N_1} r^n \le \sum_{n=-\infty}^{N_1} r^n = r^{N_1+1}/(r-1)$, which is at most $\frac{r}{r-1}$ times the total movement of server x, because the latter is at least r^{N_1}.

To bound the relocating cost, say x exits M_n at p and re-enters it at p'. Then p and p' are at the boundary of M_n and M_{n+u} for $u \in \{-1, +1\}$. Let b be the distance traveled by x in M_{n+u} between the times when it is entered at p and when it is next exited. If this exiting is at p', then the relocating cost $d(p, p')$ is at most b by the triangle inequality. Otherwise, x exits M_{n+u} at a point p'' at the boundary of M_{n+u} and M_{n+2u}. If $u = 1$, then $d(p, p') \le d(s, p) + d(s, p') = 2r^{n+1}$ and $b \ge d(p, p'') \ge d(s, p'') - d(s, p) = r^{n+2} - r^{n+1} = (r-1)r^{n+1}$. If $u = -1$, then $d(p, p') \le d(s, p) + d(s, p') = 2r^n$ and $b \ge d(p, p'') \ge d(s, p) - d(s, p'') = r^n - r^{n-1} = \frac{r-1}{r} r^n$. In both cases, the relocating cost $d(p, p')$ is at most $\frac{2r}{r-1} b$. Thus, the total relocating cost of all servers x_n is at most $\frac{2r}{r-1}$ times the total distance traveled by x.

Thus, the sum of deployment, tracking and relocating cost of the servers x_n is at most $\frac{4r-1}{r-1}$ times the distance traveled by x. This shows (5), giving the statement of the theorem. ◀

The last theorem can also be slightly generalized to the case where instead of *strict* ρ-competitiveness, an additive term proportional to r^n is allowed. It is not difficult to show the following specialization for the line, where the premise can be weakened to require competitiveness only on a single interval:

▶ **Corollary 17.** *Let $0 < a < b$. The infinite server problem is competitive on the line if and only if it is competitive on $(\{0\} \cup [a, b], 0)$.*

6 Open Problems

The most obvious open problem is whether the infinite server problem is competitive on general metric spaces. A challenging special case is to resolve the question for the real line. Similarly, improving the MOO algorithm and settling the question for layered graphs remains open. It would also be interesting to find a metric space with a competitive ratio greater than 3.146 for the infinite server problem or the (h, k)-server problem when $k \gg h$. Another possible line of research is to consider randomized algorithms.

References

1 Nikhil Bansal, Marek Eliáš, Łukasz Jeż, and Grigorios Koumoutsos. The (h, k)-server problem on bounded depth trees. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 2017*, pages 1022–1037. SIAM, 2017. doi:10.1137/1.9781611974782.65.

2 Nikhil Bansal, Marek Eliáš, Łukasz Jeż, Grigorios Koumoutsos, and Kirk Pruhs. Tight bounds for double coverage against weak adversaries. In *International Workshop on Approximation and Online Algorithms*, pages 47–58. Springer, 2015. doi:10.1007/978-3-319-28684-6_5.

3 Yair Bartal and Eddie Grove. The harmonic k-server algorithm is competitive. *J. ACM*, 47(1):1–15, January 2000. doi:10.1145/331605.331606.

4 Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theoretical Computer Science*, 324(2):337–345, 2004. doi:10.1016/j.tcs.2004.06.001.

5 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, NY, USA, 1998.

6 Marek Chrobak, Howard Karloff, Tom Payne, and Sundar Vishwanathan. New results on server problems. *SIAM J. Discret. Math.*, 4(2):172–181, March 1991. doi:10.1137/0404017.

7 Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.*, 20(1):144–148, February 1991. doi:10.1137/0220008.

8 Marek Chrobak and Lawrence L. Larmore. The server problem and on-line games. In *On-line Algorithms, volume 7 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Citeseer, 1992.

9 Ilan Reuven Cohen, Alon Eden, Amos Fiat, and Łukasz Jeż. Pricing online decisions: Beyond auctions. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 73–91, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973730.7.

10 János Csirik, Csanád Imreh, John Noga, Steven S. Seiden, and Gerhard J. Woeginger. Buying a constant competitive ratio for paging. In *Proceedings of the 9th Annual European Symposium on Algorithms*, ESA'01, pages 98–108, London, UK, 2001. Springer-Verlag. doi:10.1007/3-540-44676-1_8.

11 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. On the additive constant of the k-server work function algorithm. In *International Workshop on Approximation and Online Algorithms*, pages 128–134. Springer, 2009. doi:10.1007/978-3-642-12450-1_12.

12 Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 454–463 vol.2, Oct 1990. doi:10.1109/FSCS.1990.89566.

13 Sandy Irani and Ronitt Rubinfeld. A competitive 2-server algorithm. *Information Processing Letters*, 39(2):85–91, 1991. doi:10.1016/0020-0190(91)90160-J.

14 Kamal Jain. Personal Communication.

15 Elias Koutsoupias. Weak adversaries for the k-server problem. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS'99, Washington, DC, USA, 1999. IEEE Computer Society. doi:10.1109/SFFCS.1999.814616.

16 Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, May 2009. doi:10.1016/j.cosrev.2009.04.002.

17 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, September 1995. doi:10.1145/210118.210128.

18 Elias Koutsoupias and Christos H. Papadimitriou. The 2-evader problem. *Information Processing Letters*, 57(5):249–252, March 1996. doi:10.1016/0020-0190(96)00010-5.

19 Mark Manasse, Lyle McGeoch, and Daniel Sleator. Competitive algorithms for on-line problems. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC'88, pages 322–333, New York, NY, USA, 1988. ACM. doi:10.1145/62212.62243.

20 Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, February 1985. doi:10.1145/2786.2793.

21 Neal Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994. doi:10.1007/BF01189992.

Quantum Automata Cannot Detect Biased Coins, Even in the Limit*

Guy Kindler[1] and Ryan O'Donnell[2]

1 **School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel**
gkindler@cs.huji.ac.il

2 **Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA**
odonnell@cs.cmu.edu

Abstract

Aaronson and Drucker (2011) asked whether there exists a quantum finite automaton that can distinguish fair coin tosses from biased ones by spending significantly more time in accepting states, on average, given an infinite sequence of tosses. We answer this question negatively.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases quantum automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.15

1 Introduction

In a 2011 work, Aaronson and Drucker [2] investigated the ability of a finite automaton to distinguish, given an infinite sequence of coin tosses, whether the coins are fair or $(\frac{1}{2} \pm \epsilon)$-biased. There are several axes of consideration discussed in [2], three of which we state here:

1. Whether the automaton is classical (and probabilistic), or quantum.
2. Whether $\epsilon > 0$ is "known" or not; i.e., whether the automaton can depend on ϵ.
3. The mechanism by which the automaton makes its decision. One possibility is that the automaton guesses "biased" by halting, and guesses "fair" by running forever. A laxer possibility is that the automaton always runs forever, with each of its states designated "biased" or "fair"; its final decision is based on the limiting time-average it spends in "biased" vs. "fair" states. We refer to the two mechanisms as "one-sided halting" and "limiting acceptance".

For example, an old result of Hellman and Cover [5] is that even when ϵ is known and limiting acceptance is allowed, a classical automaton needs $\Omega(1/\epsilon)$ states to solve the problem. On the other hand, Aaronson and Drucker made the interesting observation that for every fixed known ϵ, there's a quantum automaton with just 2 states that solves the problem using one-sided halting. They also showed no quantum automaton with a fixed number of states can solve the problem for every *unknown* ϵ, if the decision mechanism is one-sided halting.

Aaronson and Drucker asked whether the same negative result holds even if the automaton is allowed to use the limiting acceptance decision mechanism. Indeed, for the 48 different variations of the problem they considered, this was the only variant that remained unsolved.

* Supported by NSF grant CCF-1618679 and by BSF grant 2012220.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 15; pp. 15:1–15:8

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

In 2014, Aaronson called this question one of the "Ten Most Annoying Problems in Quantum Computing" [1].

In this work, we make the world of quantum computing 10% less annoying by resolving the problem in the negative. Stated informally, our main theorem is the following (a precise phrasing appears below after we give some formal definitions):

▶ **Theorem 1.** *There is no quantum finite automaton that has the following property, simultaneously for every* $\epsilon \in [-\frac{1}{2}, \frac{1}{2}] \setminus \{0\}$*: Given access to an infinite sequence of coin tosses, if the coin is* $(\frac{1}{2} + \epsilon)$*-biased then the automaton spends at least* $2/3$ *of its time guessing "biased", and if the coin is fair then the automaton spends at least* $2/3$ *of its time guessing "fair".*

Proving this theorem involves a careful understanding of the fixed points of quantum channels.

2 Classical and quantum automata

In this section we review the definitions of probabilistic and quantum finite state automata. Although we are ultimately only concerned with quantum automata, we feel it is instructive to also discuss probabilistic automata at the same time. All of our automata will have input alphabet $\Sigma = \{0, 1\}$, which may be thought of as $\{\text{tails}, \text{heads}\}$.

A classical deterministic automaton on alphabet $\Sigma = \{0, 1\}$ has some d *basic-states*,[1] an initial basic-state $i_0 \in [d]$, and transition rules $f_0, f_1 : [d] \to [d]$. Given a sequence of input symbols $w_1, w_2, w_3, \cdots \in \{0, 1\}$, the automaton operates as follows: It starts in basic-state i_0 at time 0. Then, if it is in basic-state i_t at time $t \in \mathbb{N}$, it transitions to basic-state $f_{w_{t+1}}(i_t)$ at time $t+1$. Automata also typically have their basic-states classified as "accept" or "reject"; we discuss this more later.

One can also consider classical *probabilistic* automata. These have randomized transitions, which can be encoded by a pair of $d \times d$ *stochastic matrices* S_0, S_1. Now at any time t the automaton can be in a "probabilistic-state", represented by a length-d probability vector π_t. (An initial probabilistic-state π_0 is also specified.) On reading symbol w_{t+1}, the automaton transitions to the probabilistic-state $\pi_{t+1} = S_{w_{t+1}} \pi_t$.

Finally, the setting for a *quantum* automaton is a d-dimensional Hilbert space $\mathcal{H}$ (which we may think of as having an orthonormal basis of "basic-state vectors" $|1\rangle, \ldots, |d\rangle$). At any time t, the automaton has a "quantum-state", which is a density operator $\rho_t \in \mathcal{B}(\mathcal{H})$. Here $\mathcal{B}(\mathcal{H})$ denotes the set of linear operators on $\mathcal{H}$, and a density operator means a positive semidefinite operator of trace 1. (Probabilistic-states are the special case of quantum-states in which ρ_t is diagonal with respect to $|1\rangle, \ldots, |d\rangle$.) The transition rules are now any two allowable quantum transformations Φ_0, Φ_1; i.e., they are *quantum channels (superoperators)* on $\mathcal{B}(\mathcal{H})$. Here a quantum channel means a linear map $\Phi : \mathcal{B}(\mathcal{H}) \to \mathcal{B}(\mathcal{H})$ that is completely positive and trace-preserving; an equivalent condition is that there exist (non-unique) *Kraus operators* $K_1, \ldots, K_r \in \mathcal{B}(\mathcal{H})$ with $\sum_{i=1}^r K_i^\dagger K_i = \mathbb{1}$ such that $\Phi(\rho) = \sum_{i=1}^r K_i \rho K_i^\dagger$. (For more on quantum channels, see e.g. [7].) Again, an initial quantum-state ρ_0 is given, and on reading symbol w_{t+1}, the automaton transitions from quantum-state ρ_t to quantum-state $\rho_{t+1} = \Phi_{w_{t+1}}(\rho_t)$.

[1] There is an unfortunate terminology clash involving the word "state" – in automata theory, "states" are the basic vertices in automaton graphs, whereas in quantum mechanics a "state" usually means the "mixed quantum state" or "density operator" of a given system. Throughout we'll refer to the former as "basic-states" and the latter as "quantum-states".

Automata with random inputs

This paper is concerned with automata whose inputs are infinite sequences of p-biased coin tosses, $p \in [0,1]$. More formally, we always assume the input symbols $w_1, w_2, w_3, \cdots \in \{0,1\}$ are chosen independently at random with $\mathbf{Pr}[w_t = 1] = p$. Because of this assumption, we can give a simplified formalization of probabilistic and quantum automata. In the case of probabilistic automata, at each time step (independently) we apply S_1 with probability p and S_0 with probability $1-p$. It is clear that this is equivalent to simply applying the stochastic matrix $S_p \coloneqq pS_1 + (1-p)S_0$ at each time step. In other words, the probabilistic-state of a probabilistic automaton after t time steps is simply $S_p^t \pi_0$. The setup is precisely equivalent to a Markov chain on $[d]$ with transition matrix S_p.

Similarly for quantum automata, at each time step we apply Φ_1 with probability p and Φ_0 with probability $1-p$; this is physically equivalent to simply applying the channel $\Phi_p \coloneqq p\Phi_1 + (1-p)\Phi_0$ at each time step. (This is ultimately because being in quantum-state ρ with probability p and quantum-state ρ' with probability $1-p$ is physically equivalent to being in quantum-state $p\rho + (1-p)\rho'$.) Thus the quantum-state of a probabilistic automaton after t time steps is simply $\Phi_p^t(\rho_0)$; we have here the quantum analogue of a Markov chain.

Automaton acceptance probability

As discussed in Section 1, we will be considering "limiting acceptance", the most relaxed possible notion for automaton acceptance. We first define this in the context of probabilistic automata. Here, each basic-state in $[d]$ is classified as either guessing "Fair" or "Biased". We write $e_{\text{fair}} \in \mathbb{R}^d$ for the 0-1 indicator of the Fair states. Thus if the automaton is in probabilistic-state $\pi \in \mathbb{R}^d$, the probability it is in a Fair basic-state is $\langle e_{\text{fair}}, \pi \rangle$. We then consider, for a sequence of T coin tosses, the *average* probability with which the automaton is in a Fair basic-state:

$$f_T(p) \coloneqq \frac{1}{T}\sum_{t=1}^{T} \langle e_{\text{fair}}, S_p^t \pi_0 \rangle = \Big\langle e_{\text{fair}}, \Big(\frac{1}{T}\sum_{t=1}^{T} S_p^t\Big)\pi_0 \Big\rangle.$$

Finally, we consider the limiting value of this probability:

$$f(p) \coloneqq \lim_{T\to\infty} f_T(p) = \langle e_{\text{fair}}, S_p^\infty \pi_0 \rangle, \quad \text{where } S_p^\infty \coloneqq \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} S_p^t.$$

Here we relied on the well-known fact that the limiting matrix S_p^∞ exists. (In fact, S_p^∞ is also a stochastic matrix, and it acts by projection onto the 1-eigenspace of S_p; we discuss this further in Section 3.) One may then say that the probabilistic automaton "guesses Fair in the limit" if $f(p) \geq \frac{2}{3}$, and "guesses Biased in the limit" if $f(p) \leq \frac{1}{3}$. (It may be considered "indecisive" otherwise.)

The definitions for a quantum automaton are extremely similar. The automaton is assumed to come equipped with an "acceptance POVM", $\{E_{\text{fair}}, \mathbb{1} - E_{\text{fair}}\}$. (Here $E_{\text{fair}} \in \mathcal{B}(\mathcal{H})$ is any operator satisfying $0 \preceq E_{\text{fair}} \preceq \mathbb{1}$, and $\mathbb{1}$ denotes the identity operator.) If the automaton is in quantum-state ρ, the probability of it measuring "Fair" is $\langle E_{\text{fair}}, \rho \rangle \coloneqq \text{tr}(E_{\text{fair}}^\dagger \rho)$. We can then again define the limiting average probability of guessing "Fair" via

$$f_T(p) \coloneqq \frac{1}{T}\sum_{t=1}^{T} \langle E_{\text{fair}}, \Phi_p^t \pi_0 \rangle = \Big\langle E_{\text{fair}}, \Big(\frac{1}{T}\sum_{t=1}^{T} \Phi_p^t\Big)\pi_0 \Big\rangle,$$

$$f(p) \coloneqq \lim_{T\to\infty} f_T(p) = \langle E_{\text{fair}}, \Phi_p^\infty \pi_0 \rangle, \quad \text{where } \Phi_p^\infty \coloneqq \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \Phi_p^t. \tag{1}$$

Again, it is known that the limiting operator Φ_p^∞ exists; this is explicitly discussed in Section 3. As before, one may say that the quantum automaton "guesses Fair in the limit" if $f(p) \geq \frac{2}{3}$, and "guesses Biased in the limit" if $f(p) \leq \frac{1}{3}$.

We may now state the main theorem of this paper:

▶ **Theorem 2.** *In the setting of quantum automata reading p-biased bits (as described above), the function f from* (1) *is a continuous function of $p \in (0,1)$.*

This theorem is a formal strengthening of Theorem 1, our negative result for coin distinguishing stated in Section 1. For example, it implies that if an automaton guesses "Fair" in the limit" for $p = \frac{1}{2}$, then for all sufficiently small ϵ it *cannot* guess "Biased" in the limit for $p = \frac{1}{2} \pm \epsilon$. In fact, we get the inability of quantum automata to distinguish p-biased and $(p \pm \epsilon)$-biased coins with limiting acceptance for any fixed $p \in (0,1)$. As noted in [2], this is sharp in the sense that there is a trivial 2-state deterministic classical automaton that distinguishes a 0-biased coin from any ϵ-biased coin, even with one-sided halting.

3 Outline of the proof

Here we give an outline of the proof of Theorem 2. At the same time, it will be instructive to outline the analogous proof in the special case of probabilistic automata. To prove that the limiting acceptance probability $f(p)$ from (1) is continuous for $p \in (0,1)$, it is enough to prove the following:

▶ **Theorem 3.** *Φ_p^∞ is continuous for $p \in (0,1)$.*

Here for definiteness we can take the metric on channels induced by the operator norm on $\mathcal{B}(\mathcal{H})$; Theorem 2 then follows because matrix multiplication and inner product are continuous.

Now is a good time to review the properties of Φ_p^∞. In general, let Φ denote any channel on $\mathcal{B}(\mathcal{H})$. Then the following are known [7, Prop. 6.3] (and easy) facts: First, $\Phi^\infty := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \Phi^t$ exists and is itself a channel. Second, as an operator, Φ^∞ acts as projection onto the fixed points $V_1(\Phi)$ of Φ. Here we are using the following notation:

▶ **Notation 4.** *For any operator A we write $V_1(A)$ for the eigenspace of A with eigenvalue* 1*, i.e., the* invariant *subspace for A.*

As mentioned earlier, the analogous statements are true regarding S^∞, when S is a stochastic operator. (In both the probabilistic and quantum cases, the essential point is that the operator in question has spectral radius 1.)

Returning to Theorem 3, certainly $\Phi_p = p\Phi_1 + (1-p)\Phi_0$ varies continuously for $p \in [0,1]$. But what we need to prove is that the *invariant subspace* $V_1(\Phi_p)$ of Φ_p varies continuously for $p \in (0,1)$. There is one obvious potential obstruction: the *dimension* of $V_1(\Phi_p)$ might change as p varies. (As we will see, this is actually the only obstruction.) Now in general, slightly perturbing a matrix *can* change the dimension of its 1-eigenspace. However we are not concerned with completely general perturbations: we are just considering all the convex combinations of two fixed channels Φ_0, Φ_1. The main technical theorem in our paper will be the following:

▶ **Theorem 5.** *For any channels Φ_0, Φ_1, the dimension $\dim V_1(\Phi_p)$ is the same for all $p \in (0,1)$.*

We will discuss the intuition for this theorem below. But first we will observe that Theorem 3 is an elementary linear-algebraic consequence of Theorem 5. This deduction of Theorem 3 from Theorem 5 is a little more familiar if we consider $\mathbb{1} - \Phi_p$ rather than Φ_p. Then Φ_p^∞ is the projection onto the kernel of $\mathbb{1} - \Phi_p$, and it is elementary that, given a continuously-parameterized family of matrices like $p \mapsto \mathbb{1} - \Phi_p$, the kernel varies continuously wherever the nullity (in this case, $\dim V_1(\Phi_p)$) is locally constant. For a simple explicit proof see, e.g., [6].

Thus all that remains in this work is to prove Theorem 5. We will do this in Section 4, but first we provide some intuition and introduce a key definition, that of *combinatorially equivalent channels*.

3.1 Intuition for Theorem 5

All of our discussion so far applies equally to probabilistic automata defined by stochastic matrices S_0, S_1. So let us first consider the analogue of Theorem 5 in this case. Here we have a family of Markov chains defined by $S_p = pS_1 + (1-p)S_0$ and we want to consider the dimension of their invariant subspaces. It is well known that the invariant subspace $V_1(S)$ of the Markov chain defined by S is spanned by a linearly independent set of invariant *probabilistic-states*. Thus $\dim V_1(S)$ is equal to the number of linearly independent ("fundamentally different", one might say) invariant distributions.

In the study of Markov chains, it's popular to focus on the irreducible case, in which case there is a unique invariant probability distribution; i.e., $\dim V_1(S) = 1$. However in general we must consider reducible Markov chains (the "mathematically annoying case", as Hellman and Cover [5] put it). Fortunately, the theory of reducible Markov chains is well developed, and it is known that there is one linearly independent invariant distribution per every *communication class* of the Markov chain. Here the "communication classes" of the Markov chain defined by S are precisely the strongly connected components of the underlying digraph on $[d]$; i.e., the graph which has a directed edge (i, j) whenever $S_{ij} \neq 0$. Given this theory, it is easy to deduce the analogue of Theorem 5; the point is that for any fixed S_0, S_1, *the underlying digraph of S_p is the same for all $p \in (0, 1)$*. Since $S_p = pS_1 + (1-p)S_0$, an edge (i, j) is present in S_p if and only if it is present in either S_0 or S_1. Thus S_p has the same set (hence number) of communication classes for all $p \in (0, 1)$, as needed.

In this paper, we show there is an analogous sequence of ideas in the quantum case, using some of the recently developed theory of fixed points of quantum channels. Given a quantum channel Φ, it is known [7, Cor. 6.5] that $V_1(\Phi)$ is always spanned by linearly independent *quantum-states*. The analogous notion to communication classes is that of *minimal enclosures*. Further, similar to how the communication classes of a Markov chain are determined only by the nonzero pattern of its transition matrix, the minimal enclosures of a quantum channel are determined only by its Kraus operators. We then make use of the fact that all the convex combinations Φ_p of two channels Φ_0, Φ_1 have related Kraus operators. Specifically, we introduce the following notion:

▶ **Definition 6.** We will say that two channels Φ and $\widehat{\Phi}$ (with the same Hilbert space $\mathcal{H}$) are *combinatorially equivalent* if there are Kraus operators $K_1, \dots, K_r$ for Φ and $\widehat{K}_1, \dots, \widehat{K}_{\widehat{r}}$ for $\widehat{\Phi}$ such that each K_i is proportional to some $\widehat{K}_{i'}$ and vice versa.

Given channels Φ_0, Φ_1 with Kraus operators $\{K_i^{(0)} : i \in [r_0]\}, \{K_j^{(1)} : j \in [r_1]\}$ respectively, the channel $\Phi_p = p\Phi_1 + (1-p)\Phi_0$ has Kraus operators $\{\sqrt{1-p}K_i^{(0)} : i \in [r_0]\} \cup \{\sqrt{p}K_j^{(1)} : j \in [r_1]\}$. Thus the channels Φ_p are all pairwise combinatorially equivalent for $p \in (0, 1)$

(though not necessarily for $p \in \{0, 1\}$). To show Theorem 5, it therefore suffices to show the following more general result:

▶ **Theorem 7.** *Suppose* Φ *and* $\widehat{\Phi}$ *are combinatorially equivalent. Then* $\dim V_1(\Phi) = \dim V_1(\widehat{\Phi})$.

4 The last step: proof of Theorem 7

To prove Theorem 7, we use some known results concerning the decomposition of a quantum channel into irreducible components, and the structure of its invariant quantum-states. We will specifically use the key decomposition theorem appearing variously as [7, Theorem 6.14], [3, Theorem 7], [4, Theorem 7.2].

Let Φ denote a quantum channel on $\mathcal{B}(\mathcal{H})$ with Kraus operators $K_1, \ldots, K_r$. We are interested in $m = \dim V_1(\Phi)$, the dimension of the space of Φ's fixed points. As Φ is a quantum channel, it is known [7, Prop. 6.1] that its spectral radius is 1 and that it has at least one eigenvalue equal to 1; thus $m \geq 1$. As mentioned, it is also known [7, Cor. 6.5] that $V_1(\Phi)$ is always spanned by some m linearly independent *quantum-states*.

If ρ is a quantum-state, its *support* $\mathrm{supp}(\rho)$ is simply the range of ρ as a subspace of $\mathcal{H}$. The *recurrent subspace* for Φ is the subspace of $\mathcal{H}$ defined by

$$\mathcal{R} = \mathrm{span}\{\mathrm{supp}(\rho) : \rho \text{ is an invariant quantum-state}\}.$$

The orthogonal complement of $\mathcal{R}$ in $\mathcal{H}$ is denoted $\mathcal{D}$; this is the *decaying* (or *transient*) subspace. A subspace $\mathcal{V} \subseteq \mathcal{H}$ is called an *enclosure* if $\mathrm{supp}(\rho) \subseteq \mathcal{V} \implies \mathrm{supp}(\Phi(\rho)) \subseteq \mathcal{V}$ for all quantum-states ρ. We can relate this concept to Kraus operators via the following equivalence:

▶ **Fact 8** ([4, Proposition 4.4]). *$\mathcal{V}$ is an enclosure if and only if $K_i\mathcal{V} \subseteq \mathcal{V}$ for all Kraus operators K_i.*

An enclosure $\mathcal{V}$ is called *minimal* if it is nonzero and all enclosures $\mathcal{V}' \subseteq \mathcal{V}$ are equal to either $\{0\}$ or $\mathcal{V}$. It is also known [3, Prop. 15] that a subspace of $\mathcal{H}$ is a minimal enclosure if and only if it is the support of an extremal invariant quantum-state, meaning one that cannot be written as a nontrivial convex combination of two distinct invariant quantum-states. One consequence is that

$$\begin{aligned}\mathcal{R} &= \mathrm{span}\{\mathrm{supp}(\rho) : \rho \text{ is an extremal invariant quantum-state}\} \\ &= \mathrm{span}\{\mathcal{V} : \mathcal{V} \text{ is a minimal enclosure}\}. \end{aligned} \tag{2}$$

The theorems [7, Theorem 6.14], [3, Theorem 7], [4, Theorem 7.2] characterize $V_1(\Phi)$ and the quantum-states therein in slightly different ways. To explain, we make some definitions.

▶ **Definition 9.** (In this definition, k, $m_1, \ldots, m_k$, $d_1, \ldots, d_k$ denote positive integers.)

Given Φ, we define a *minimal enclosure decomposition* to be an orthogonal decomposition of $\mathcal{H}$ into subspaces

$$\mathcal{H} = \mathcal{D} \oplus \bigoplus_{i=1}^{k} \mathcal{W}_i, \quad \text{where } \mathcal{W}_i = \bigoplus_{j=1}^{m_i} \mathcal{V}_{i,j} \tag{3}$$

for which the following properties hold:

1. $\mathcal{D}$ is the decaying subspace for Φ.
2. Each $\mathcal{V}_{i,j}$ is a minimal enclosure.
3. Each $\dim \mathcal{V}_{i,j} = d_i$ for all $1 \le j \le m_i$.
4. For any minimal enclosure $\mathcal{X}$ of Φ and any $1 \le i \le k$, if $\mathcal{X}$ is not orthogonal to $\mathcal{W}_i$ then $\mathcal{X} \subseteq \mathcal{W}_i$. (In particular, if $m_i = 1$ then $\mathcal{X}$ must equal $\mathcal{W}_i$.)
5. The decomposition (3) is maximal, in the sense that it is not possible to increase k.

▶ Remark. In fact, one can show there is always a *unique* minimal enclosure decomposition. However, we have not found this exact statement appearing in the literature, and in this paper we will prefer to simply cite known results.

▶ **Definition 10.** Suppose we have a minimal enclosure decomposition for Φ as above. Fix any ordered orthogonal basis for $\mathcal{H}$ compatible with (3) (meaning the first $\dim \mathcal{D}$ elements span $\mathcal{D}$, the next $m_1 d_1$ elements come in m_1 groups of d_1 spanning $\mathcal{V}_{1,1}, \dots, \mathcal{V}_{1,m_1}$ respectively, etc.). Let $X \in \mathcal{B}(\mathcal{H})$, and think of X in its matrix form with respect to the ordered basis.

Then we say that X *respects the minimal enclosure decomposition* if X is block-diagonal with blocks corresponding to $\mathcal{D}, \mathcal{W}_1, \dots, \mathcal{W}_k$, and furthermore X is 0 on the $\mathcal{D}$-block and is of the form $A_i \otimes \rho_i$ on the $\mathcal{W}_i$-block for some $A_i \in \mathbb{C}^{m_i \times m_i}$ and some strictly positive density matrix $\rho_i \in \mathbb{C}^{d_i \times d_i}$. In symbols,

$$X = 0 \oplus \bigoplus_{i=1}^{k} A_i \otimes \rho_i.$$

(We remark that the property of respecting the minimal enclosure decomposition does not depend on the choice of the compatible orthogonal basis.)

In combination, [7, Theorem 6.14], [3, Theorem 7] state the following:[2]

▶ **Theorem 11.** *Given any channel Φ, there exists a minimal enclosure decomposition as in* (3) *such that $V_1(\Phi)$ consists precisely of* all *$X \in \mathcal{B}(\mathcal{H})$ that respect the decomposition. (An immediate consequence is that $m = \dim V_1(\Phi) = \sum_i m_i^2$.) Finally, the quantum-states that are invariant are precisely all such X with $A_i = \lambda_i \sigma_i$, where $\sigma_1, \dots, \sigma_k$ are density matrices and $\lambda_1, \dots, \lambda_k$ are nonnegative reals summing to* 1.

The statement of [4, Theorem 7.2] is slightly different:[3]

▶ **Theorem 12.** *Given any channel Φ, at least one minimal enclosure decomposition exists. Furthermore, given* any *minimal enclosure decomposition*

$$\mathcal{H} = \mathcal{D} \oplus \bigoplus_{i=1}^{\widehat{k}} \widehat{\mathcal{W}}_i, \quad \text{where } \widehat{\mathcal{W}}_i = \bigoplus_{j=1}^{\widehat{m}_{\widehat{k}}} \widehat{\mathcal{V}}_{i,j},$$

every invariant quantum-state for Φ respects it. (As an immediate consequence, we have $m = \dim V_1(\Phi) \le \sum_i \widehat{m}_i^2$.)

[2] [7] deals with the invariant subspace whereas [3] deals with the invariant quantum-states. The fact that the ρ_i's are strictly positive is in [7]. Finally, [3] does not explicitly show that the minimal enclosure decomposition satisfies condition (4) in Definition 9. However, it's implicit and it's easy to deduce: we know that any minimal enclosure $\mathcal{X}$ is the support of some extremal invariant quantum-state ρ, and it's clear that if this support is not entirely within a single $\mathcal{W}_i$-block then ρ would not be extremal.

[3] The first statement of this theorem is [4, Proposition 7.1], except that that Proposition does not include either condition (4) of Definition 9 for those i with $m_i = 1$. However it is evident from the proof that this is an oversight; a personal communication from the authors confirmed this. Also, [4, Proposition 7.1] does not explicitly state condition (5) of Definition 9, but it is obtained by the proof, and is in fact needed for correctness of the proof.

We are now able to give the proof of Theorem 7.

Proof of Theorem 7. Write $m = \dim V_1(\Phi)$ and $\widehat{m} = \dim V_1(\widehat{\Phi})$. Since Φ and $\widehat{\Phi}$ play symmetric roles, it suffices to show $\widehat{m} \leq m$. Apply Theorem 11 to Φ, obtaining a minimal enclosure decomposition as in (3). We have $m = \sum_{i=1}^{k} m_i^2$. We claim that this decomposition is also a minimal enclosure decomposition for $\widehat{\Phi}$. This will finish the proof of $\widehat{m} \leq m$, by Theorem 12.

To see the claim, we first observe that every enclosure $\mathcal{V}$ for Φ is an enclosure for $\widehat{\Phi}$ (and vice versa). This follows from Fact 8: $\mathcal{V}$ satisfies $K_i\mathcal{V} \subseteq \mathcal{V}$ for each Kraus operator K_i of Φ, and hence the same is true for the Kraus operators $\widehat{K}_{i'}$ of $\widehat{\Phi}$, by combinatorial equivalence of Φ and $\widehat{\Phi}$. It then follows by definition that every *minimal* enclosure for Φ is also a minimal enclosure for $\widehat{\Phi}$ (and vice versa). Finally, the claim now follows because Φ and $\widehat{\Phi}$ have the same decaying subspace (by (2)) and because Definition 9 of minimal enclosure decompositions depends *only* on which subspaces of $\mathcal{H}$ are minimal enclosures. ◀

Acknowledgment. We thank an anonymous reviewer for a very careful reading of the paper that fixed some inaccuracies.

References

1 Scott Aaronson. The NEW ten most annoying questions in quantum computing, May 2014. http://www.scottaaronson.com/blog/?p=1792.

2 Scott Aaronson and Andrew Drucker. Advice coins for classical and quantum computation. In *Proceedings of the 38th Annual International Colloquium on Automata, Languages and Programming*, pages 61–72, 2011.

3 Bernhard Baumgartner and Heide Narnhofer. The structures of state space concerning quantum dynamical semigroups. *Reviews in Mathematical Physics*, 24(2):1250001, 2012.

4 Raffaella Carbone and Yan Pautrat. Irreducible decompositions and stationary states of quantum channels. Technical Report 1507.08404, arXiv, 2015.

5 Martin Hellman and Thomas Cover. Learning with finite memory. *Annals of Mathematical Statistics*, 41:765–782, 1970.

6 user1551. Continuity of the basis of the null space, March 2015. http://math.stackexchange.com/a/1203782.

7 Michael Wolf. Quantum channels & operations: guided tour, 2012. http://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MichaelWolf/QChannelLecture.pdf.

A New Holant Dichotomy Inspired by Quantum Computation*†

Miriam Backens

School of Mathematics, University of Bristol, Bristol, UK
m.backens@bristol.ac.uk

Abstract

Holant problems are a framework for the analysis of counting complexity problems on graphs. This framework is simultaneously general enough to encompass many counting problems on graphs and specific enough to allow the derivation of dichotomy results, partitioning all problems into those which are in FP and those which are #P-hard. The Holant framework is based on the theory of holographic algorithms, which was originally inspired by concepts from quantum computation, but this connection appears not to have been explored before.

Here, we employ quantum information theory to explain existing results in a concise way and to derive a dichotomy for a new family of problems, which we call Holant$^+$. This family sits in between the known families of Holant*, for which a full dichotomy is known, and Holant^c, for which only a restricted dichotomy is known. Using knowledge from entanglement theory – both previously existing work and new results of our own – we prove a full dichotomy theorem for Holant$^+$, which is very similar to the restricted Holant^c dichotomy and may thus be a stepping stone to a full dichotomy for that family.

1998 ACM Subject Classification G.2.1 [Combinatorics] Counting Problems, F.2.m [Analysis of Algorithms and Problem Complexity] Miscellaneous

Keywords and phrases computational complexity, counting complexity, Holant, dichotomy, entanglement

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.16

1 Introduction

Quantum computation (QC) provided the inspiration for holographic algorithms [30], which in turn inspired the Holant framework [11]. While Holant problems are an area of active research, so far there appear to have been no attempts to apply knowledge from quantum information theory (QIT) or QC to their analysis. Yet, as we show in the following, QIT and QC offer promising new avenues of research into Holant problems.

The Holant framework encompasses a wide range of counting complexity problems on graphs, parameterised by sets of functions $\mathcal{F}$. Here, we consider functions of Boolean inputs taking values in the set of algebraic complex numbers. Each vertex in the graph is assigned a function from $\mathcal{F}$, with each edge incident on the vertex corresponding to an input of the function. This structure is associated with a complex number, the *Holant*, computed by multiplying the function values together and summing over all possible input assignments for each edge (for the full definition, see Section 2). The associated counting problem Holant $(\mathcal{F})$ is the following: given a graph and an assignment of functions from $\mathcal{F}$ to

* A full version of this paper is available on the arXiv [1].
† I acknowledge funding from EPSRC via grant EP/L021005/1.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 16; pp. 16:1–16:14


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

vertices, find the value of the Holant [11]. From a QIT perspective, each function can be considered as a tensor with one index for each input, making $\text{Holant}(\mathcal{F})$ the evaluation of a tensor network contraction.

The Holant framework is general enough to include problems such as counting matchings or perfect matchings, counting vertex covers [11], or counting Eulerian orientations [21]. It also encompasses other counting complexity frameworks like counting constraint satisfaction problems (#CSP) or counting graph homomorphisms [11]. On the other hand, the Holant framework is specific enough to allow the derivation of dichotomy theorems, which state that for function sets $\mathcal{F}$ within certain classes, the Holant problem is either in FP or it is #P-hard. By an analogue of Ladner's Theorem about NP-intermediate problems [23], such a dichotomy is not expected to hold for general counting problems [6].

One example of such a dichotomy is that for Holant^*. The problem $\text{Holant}^*(\mathcal{F})$ is equal to $\text{Holant}(\mathcal{F} \cup \mathcal{U})$, where $\mathcal{U}$ is the set of all unary functions [6]. Another example is the dichotomy for symmetric Holant^c, where all function sets considered must contain the unary functions pinning edges to values 0 or 1, respectively. Additionally, all functions are required to be *symmetric*, meaning their value depends only on the Hamming weight of the input [10]. Further dichotomies exist, but these, too, assume the availability of certain functions [12] or restrict the function sets, e.g. to symmetric or real-valued functions only [9, 27]. A full dichotomy for all Holant problems, as well as a full dichotomy for Holant^c, have so far remained elusive.

Here, we use knowledge from QC and QIT to make a step towards a full dichotomy for Holant^c. First, we analyse existing dichotomies in quantum terms, finding natural characterisations of the Holant^* and symmetric Holant^c dichotomies. The former can be described in terms of the entanglement classes of the allowed functions. Entanglement is a core concept in quantum theory: a quantum state of multiple systems is entangled if it cannot be written as a tensor product of states of subsystems. For states of more than two systems, there are different classes of entanglement which can be used for different QIT tasks [28]; their classification is an area of ongoing research [15, 31, 24, 25, 2]. We also find that the tractable class of *affine functions* arising in the dichotomy for symmetric Holant^c (see Section 3.2) is well-known in QIT as *stabilizer states* [20].

Motivated by this, we define a new class of Holant problems, which we call Holant^+. This class encompasses Holant problems where function sets are required to contain four specific unary functions, including the two that are available in Holant^c. In this way, Holant^+ fits between Holant^*, for which there is a full dichotomy, and Holant^c, for which there is no full dichotomy. These four unary functions enable the use of a known result from entanglement theory about producing two-system entangled states from many-system ones via projections [29, 18]: this corresponds to the ability to produce non-degenerate binary functions via gadgets. In fact, we prove an extension of that result about constructing three-qubit entangled states, or equivalently ternary functions. Using this, we derive our dichotomy theorem for Holant^+, whose tractable classes are very similar to those of the dichotomy for symmetric Holant^c [11]. Our dichotomy is the first full Holant dichotomy with no restrictions on the type of functions and where only a finite number of functions are assumed available, except for the dichotomy for $\#\text{R}_3$-CSP [12].

In the following, Section 2 contains a more detailed introduction to the Holant problem and associated concepts. In Section 3, we recap the relevant existing dichotomies and results. The quantum perspective on Holant problems, together with important notions from entanglement theory, is introduced in Section 4. We define and motivate the new family of Holant problems, called Holant^+, and prove the dichotomy theorem in Section 5.

2 Holant problems

Holant problems are a framework for counting complexity problems on graphs, introduced by Cai *et al.* [11], and based on the theory of holographic algorithms developed by Valiant [30]. Let $\mathcal{F}$ be a set of complex-valued functions with Boolean inputs, also called *signatures*, and let $G = (V, E)$ be an undirected graph with vertices V and edges E. Throughout, graphs are allowed to have parallel edges and self-loops. All complex numbers are assumed to be algebraic [7]. A *signature grid* is a tuple $\Omega = (G, \mathcal{F}, \pi)$ where π is a function that assigns to each n-ary vertex $v \in V$ a function $f_v : \{0,1\}^n \to \mathbb{C}$ in $\mathcal{F}$, specifying which edge corresponds to which input. The Holant for a signature grid Ω is:

$$\text{Holant}_\Omega = \sum_{\sigma : E \to \{0,1\}} \prod_{v \in V} f_v(\sigma|_{E(v)}), \tag{1}$$

where σ is an assignment of Boolean values to each edge and $\sigma|_{E(v)}$ is the restriction of σ to the edges incident on v.

▶ **Definition 1.** The Holant problem for a set of signatures $\mathcal{F}$, denoted by $\text{Holant}(\mathcal{F})$, is defined as follows:

- *Input*: a signature grid $\Omega = (G, \mathcal{F}, \pi)$ over the signature set $\mathcal{F}$,
- *Output*: Holant_Ω.

A *symmetric signature* is a function that depends only on the Hamming weight of the input. An n-ary symmetric signature is often written as $f = [f_0, f_1, \ldots, f_n]$, where f_k is the value f takes on inputs of Hamming weight k for $k \in \{0, \ldots, n\}$. A signature is called *degenerate* if it is a product of unary signatures. Any signature that cannot be expressed as a product of unary signatures is called *non-degenerate*. Multiplying a signature by a non-zero constant does not change the complexity of evaluating the Holant, so we will usually identify functions that are equal up to non-zero scalar factor.

Given a bipartite graph, we can define a *bipartite signature grid* by specifying two signature sets $\mathcal{F}$ and $\mathcal{G}$ and assigning signatures from $\mathcal{F}$ ($\mathcal{G}$) to vertices from the first (second) partition. A bipartite signature grid is denoted by a tuple $(G, \mathcal{F} \mid \mathcal{G}, \pi)$. The corresponding *bipartite Holant problem* is $\text{Holant}(\mathcal{F} \mid \mathcal{G})$. Any signature grid can be made bipartite by inserting a new vertex carrying the binary equality signature in the middle of each edge.

2.1 Signature grids in terms of vectors

As noted in [8], any signature $f : \{0,1\}^n \to \mathbb{C}$ can be considered as a complex vector of 2^n components indexed by $\{0,1\}^n$. Let $\{|x\rangle\}_{x \in \{0,1\}^n}$ be an orthonormal basis[1] for $\mathbb{C}^{2^n}$. The vector corresponding to the signature f is then denoted by $|f\rangle = \sum_{x \in \{0,1\}^n} f(x) |x\rangle$.

Suppose $\Omega = (G, \mathcal{F} \mid \mathcal{G}, \pi)$ is a bipartite signature grid, where $G = (V, W, E)$ has vertex partitions V and W. Then the Holant for Ω can be written as:

$$\text{Holant}_\Omega = \left(\bigotimes_{w \in W} (|g_w\rangle)^T \right) \left(\bigotimes_{v \in V} |f_v\rangle \right) = \left(\bigotimes_{v \in V} (|f_v\rangle)^T \right) \left(\bigotimes_{w \in W} |g_w\rangle \right), \tag{2}$$

where the tensor products are assumed to be ordered such that, in each inner product, two systems associated with the same edge meet.

[1] In using this notation for vectors, called *Dirac notation* and common in QC and QIT, we anticipate the interpretation of the vectors associated with signatures as quantum states, cf. Section 4.

2.2 Reductions

Holographic transformations are the origin of the name 'Holant problems'. Let M be a 2 by 2 complex matrix. For any $f : \{0,1\}^n \to \mathbb{C}$, write $M \circ f$ for the function corresponding to the vector $M^{\otimes n} |f\rangle$. Furthermore, for any signature set $\mathcal{F}$, write $M \circ \mathcal{F} := \{M \circ f \mid f \in \mathcal{F}\}$.

▶ **Theorem 2** (Valiant's Holant Theorem, [30]). *Suppose $\mathcal{F}$ and $\mathcal{G}$ are two sets of signatures, M an invertible 2 by 2 complex matrix, and $\Omega = (G, \mathcal{F} \mid \mathcal{G}, \pi)$ a signature grid. Let $\Omega' = (G, M \circ \mathcal{F} \mid (M^{-1})^T \circ \mathcal{G}, \pi')$ be the signature grid resulting from Ω by replacing each f_v or g_w by $M \circ f_v$ or $(M^{-1})^T \circ g_w$, respectively. Then $\text{Holant}_\Omega = \text{Holant}_{\Omega'}$ and therefore $\text{Holant}(\mathcal{F} \mid \mathcal{G}) \equiv_T \text{Holant}\left(M \circ \mathcal{F} \mid (M^{-1})^T \circ \mathcal{G}\right)$.*

Here, $\equiv_T$ means the two problems have the same complexity. For non-bipartite signature grids, Theorem 2 implies that $\text{Holant}(\mathcal{F}) \equiv_T \text{Holant}(O \circ \mathcal{F})$, where O is any orthogonal 2 by 2 complex matrix [30]. Going from a signature set $\mathcal{F} \mid \mathcal{G}$ to $M \circ \mathcal{F} \mid (M^{-1})^T \circ \mathcal{G}$ or from $\mathcal{F}$ to $O \circ \mathcal{F}$ is a *holographic reduction*.

A *gadget* over a signature set $\mathcal{F}$ (also called $\mathcal{F}$-gate) is a fragment of a signature grid with some 'dangling' edges. Any gadget can be assigned an effective signature g. If g is the effective signature of some gadget over $\mathcal{F}$, g is said to be *realisable over* $\mathcal{F}$.

▶ **Lemma 3** ([6]). *Suppose $\mathcal{F}$ is some signature set and g is realisable over $\mathcal{F}$. Then $\text{Holant}(\mathcal{F} \cup \{g\}) \equiv_T \text{Holant}(\mathcal{F})$.*

Following [27], we define for any signature set $\mathcal{F}$, $S(\mathcal{F}) = \{g \mid g \text{ is realisable over } \mathcal{F}\}$. Then Lemma 3 implies that $\text{Holant}(S(\mathcal{F})) \equiv_T \text{Holant}(\mathcal{F})$.

If $g \notin S(\mathcal{F})$, in certain cases it is nevertheless possible to show a result like Lemma 3 by analysing a family of signature grids that differ in specific ways. This process is called *polynomial interpolation* and will not be used here, though it is a crucial ingredient in some of the results we build upon. The interested reader can find a discussion of polynomial interpolation in [11].

3 Existing results about the Holant problem

We now introduce the existing families of Holant problems and the associated dichotomy results. Gadget constructions, which are at the heart of many reductions, are easier the more signatures are known to be available. As a result, several families of Holant problems have been defined, in which certain sets of signatures are freely available – i.e. have to be included in any set $\mathcal{F}$ – and can thus be used in gadget constructions and polynomial interpolation.

3.1 Holant*

The Holant problem in which all unary signatures are freely available is $\text{Holant}^*(\mathcal{F}) = \text{Holant}(\mathcal{F} \cup \mathcal{U})$, where $\mathcal{U}$ is the set of all unary signatures [11, 6].

We begin with some definitions. Given a bit string x, let $\bar{x}$ be its bit-wise complement. Denote by $\langle \mathcal{F} \rangle$ the closure of a signature set $\mathcal{F}$ under tensor products. Furthermore, let:

- $\mathcal{T}$ be the set of all binary signatures,
- $\mathcal{E}$ the set of signatures which are non-zero only on two inputs x and $\bar{x}$, and
- $\mathcal{M}$ the set of signatures which are non-zero only on inputs of Hamming weight at most 1.

Finally, define $K = \left(\begin{smallmatrix} 1 & 1 \\ i & -i \end{smallmatrix}\right)$ and $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. The matrix K satisfies $K^T K \doteq X$, where $\doteq$ denotes equality up to non-zero scalar factor. In fact, up to multiplication by a diagonal matrix or by X itself, K is the only solution to this equation (see the full version of this paper at [1]).

▶ **Theorem 4** ([6]). *Let $\mathcal{F}$ be any set of complex valued functions in Boolean variables. The problem* $\#\mathrm{HOLANT}^*(\mathcal{F})$ *is polynomial time computable if:*

- $\mathcal{F} \subseteq \langle \mathcal{T} \rangle$, *or*
- $\mathcal{F} \subseteq \langle O \circ \mathcal{E} \rangle$, *where O is a complex orthogonal 2 by 2 matrix, or*
- $\mathcal{F} \subseteq \langle K \circ \mathcal{E} \rangle$, *or*
- $\mathcal{F} \subseteq \langle K \circ \mathcal{M} \rangle$ *or* $\mathcal{F} \subseteq \langle KX \circ \mathcal{M} \rangle$.

In all other cases, $\#\mathrm{HOLANT}^*(\mathcal{F})$ *is #P-hard. The dichotomy is still valid even if the inputs are restricted to planar graphs.*

3.2 Holant^c

$\#\mathrm{HOLANT}^c$ is the Holant problem in which only the unary signatures pinning edges to 0 or 1 are freely available [11, 10], i.e. $\#\mathrm{HOLANT}^c(\mathcal{F}) = \#\mathrm{HOLANT}(\mathcal{F} \cup \{\delta_0, \delta_1\})$ with $\delta_0 = [1, 0]$ and $\delta_1 = [0, 1]$. There is no full dichotomy for $\#\mathrm{HOLANT}^c$ yet, though there is a dichotomy that applies to sets of symmetric signatures only. This dichotomy features a new family of tractable signatures, which do not appear in the $\#\mathrm{HOLANT}^*$ dichotomy.

▶ **Definition 5.** A signature $f : \{0,1\}^n \to \mathbb{C}$ is called *affine* if it has the form:

$$f(x) = ci^{l(x)}(-1)^{q(x)}\chi_{Ax=b}(x), \tag{3}$$

where $c \in \mathbb{C}$, $i^2 = -1$, $l : \{0,1\}^n \to \mathbb{Z}_2$ is a linear function, $q : \{0,1\}^n \to \mathbb{Z}_2$ is a quadratic function, A is an m by n matrix with Boolean entries for some $0 \leq m \leq n$, $b \in \{0,1\}^m$, and χ is an indicator function which takes value 1 on inputs satisfying $Ax = b$, and 0 otherwise.

The set of all affine signatures is denoted by $\mathcal{A}$; this is already closed under tensor products. For the reader familiar with quantum information theory, the affine signatures correspond – up to a scalar factor – to stabilizer states (cf. Section 4.2).

▶ **Theorem 6** ([10]). *Let $\mathcal{F}$ be a set of complex symmetric signatures.* $\#\mathrm{HOLANT}^c(\mathcal{F})$ *is #P-hard unless $\mathcal{F}$ satisfies one of the following conditions, in which case it is in FP:*

- $\#\mathrm{HOLANT}^*(\mathcal{F})$ *is polynomial-time computable (cf. Theorem 4), or*
- *there exists a $T \in \mathcal{I}$ such that $\mathcal{F} \subseteq T \circ \mathcal{A}$, where:*

$$\mathcal{I} = \left\{ T \,\middle|\, \left(T^{-1}\right)^{\otimes T} \circ \{=_2, \delta_0, \delta_1\} \subset \mathcal{A} \right\}. \tag{4}$$

3.3 Other Holant problems

Complex-weighted Boolean #CSP (the counting constraint satisfaction problem) corresponds to a Holant problem in which equality functions of any arity are freely available. Formally, $\#\mathrm{CSP}(\mathcal{F}) = \#\mathrm{HOLANT}(\mathcal{F} \cup \mathcal{G})$, where $\mathcal{G} = \{=_1, =_2, =_3, \ldots\}$ with $=_1$ being the function that is equal to 1 on both inputs [11, 10, 12].

▶ **Theorem 7** ([12]). *Suppose $\mathcal{F}$ is a class of functions mapping Boolean inputs to complex numbers. If $\mathcal{F} \subseteq \mathcal{A}$ or $\mathcal{F} \subseteq \langle \mathcal{E} \rangle$, then* $\#\mathrm{CSP}(\mathcal{F})$ *is computable in polynomial time. Otherwise,* $\#\mathrm{CSP}(\mathcal{F})$ *is #P-hard.*

The same dichotomy also holds for $\#\mathrm{R}_3$-CSP, which corresponds to the bipartite Holant problem $\#\mathrm{HOLANT}(\mathcal{F} \mid \{=_1, =_2, =_3\})$ [12]. This dichotomy follows immediately from that for #CSP if $\mathcal{F}$ contains the binary (or indeed any non-unary) equality function, but it is non-trivial if $\mathcal{F}$ does not contain any non-unary equality functions.

In the case of $\#\mathrm{HOLANT}$ with no free signatures, there exists a dichotomy for complex-valued symmetric signatures [9] and a dichotomy for (not necessarily symmetric) signatures taking non-negative real values [27]. We will not explore those results in any detail here.

3.4 Results about ternary symmetric signatures

The hardness of problems of the form $\text{Holant}\,(\{[y_0,y_1,y_2]\} \mid \{[x_0,x_1,x_2,x_3]\})$ has been fully determined. If $[x_0,x_1,x_2,x_3]$ is degenerate, the problem is tractable by the first case of Theorem 4. If $[x_0,x_1,x_2,x_3]$ is non-degenerate, it can always be mapped to $[1,0,0,1]$ or $[1,1,0,0]$ by a holographic transformation [10]. By Theorem 2, it thus suffices to consider the cases $\{[y_0,y_1,y_2]\} \mid \{[1,0,0,1]\}$ and $\{[y_0,y_1,y_2]\} \mid \{[1,1,0,0]\}$.

There are holographic transformations which leave the signature $[1,0,0,1]$ invariant: in particular, $\left(\begin{smallmatrix}1 & 0\\ 0 & \omega\end{smallmatrix}\right) \circ [1,0,0,1] = [1,0,0,1]$ if $\omega^3 = 1$ [10]. Thus, by Theorem 2:

$$\text{Holant}\,(\{[y_0,y_1,y_2]\} \mid \{[1,0,0,1]\}) \equiv_T \text{Holant}\left(\{[y_0,\omega y_1,\omega^2 y_2]\} \mid \{[1,0,0,1]\}\right). \quad (5)$$

This relationship can be used to reduce the number of symmetric binary signatures to be considered. Following [10], a signature of the form $[y_0,y_1,y_2]$ is called *ω-normalised* if $y_0 = 0$, or there does not exist a primitive $(3t)$-th root of unity λ, where $gcd(t,3) = 1$, such that $y_2 = \lambda y_0$. Similarly, a unary signature $[a,b]$ is ω-normalised if $a = 0$, or there does not exist a primitive $(3t)$-th root of unity λ, where $gcd(t,3) = 1$, such that $b = \lambda a$.

▶ **Theorem 8** ([10]). *Let $\mathcal{G}_1, \mathcal{G}_2$ be two sets of signatures and let $[y_0,y_1,y_2]$ be a ω-normalised and non-degenerate signature. In the case of $y_0 = y_2 = 0$, further assume that $\mathcal{G}_1$ contains a unary signature $[a,b]$ which is ω-normalised and satisfies $ab \neq 0$. Then:*

$$\text{Holant}\,(\{[y_0,y_1,y_2]\} \cup \mathcal{G}_1 \mid \{[1,0,0,1]\} \cup \mathcal{G}_2) \equiv_T \#\text{CSP}(\{[y_0,y_1,y_2]\} \cup \mathcal{G}_1 \cup \mathcal{G}_2). \quad (6)$$

More specifically, $\text{Holant}\,(\{[y_0,y_1,y_2]\} \cup \mathcal{G}_1 \mid \{[1,0,0,1]\} \cup \mathcal{G}_2)$ *is #P-hard unless:*

- $\{[y_0,y_1,y_2]\} \cup \mathcal{G}_1 \cup \mathcal{G}_2 \subseteq \langle \mathcal{E} \rangle$, *or*
- $\{[y_0,y_1,y_2]\} \cup \mathcal{G}_1 \cup \mathcal{G}_2 \subseteq \mathcal{A}$,

in which cases the problem is in FP.

▶ **Theorem 9** ([10]). $\text{Holant}\,(\{[y_0,y_1,y_2]\} \mid \{[x_0,x_1,x_2,x_3]\})$ *is #P-hard unless $[y_0,y_1,y_2]$ and $[x_0,x_1,x_2,x_3]$ satisfy one of the following conditions, in which case the problem is in FP:*

- $[x_0,x_1,x_2,x_3]$ *is degenerate, or*
- *there is a 2 by 2 matrix M such that:*
  - $[x_0,x_1,x_2,x_3] = M \circ [1,0,0,1]$ *and* $M^T \circ [y_0,y_1,y_2]$ *is in* $\mathcal{A} \cup \langle \mathcal{E} \rangle$,
  - $[x_0,x_1,x_2,x_3] = M \circ [1,1,0,0]$ *and* $[y_0,y_1,y_2] = (M^{-1})^T \circ [0,a,b]$ *for some* $a,b \in \mathbb{C}$.

4 The quantum state perspective on signature grids

In Section 2.1, we introduced the idea of considering signatures as complex vectors. This perspective is useful for proving Valiant's Holant Theorem, which is at the heart of the theory of Holant problems. It also gives a connection to the theory of QC.

In QC and QIT, the basic system of interest is a *qubit* (quantum bit), which takes the place of the usual bit in standard computer science. The state of a qubit is described by a vector[2] in $\mathbb{C}^2$. State spaces compose by tensor product, i.e. the state of n qubits is described by a vector in $\left(\mathbb{C}^2\right)^{\otimes n}$, which is isomorphic to $\mathbb{C}^{2^n}$. Thus, the vector associated with an n-ary signature can be considered to be an (unnormalised) quantum state of n qubits.

[2] Strictly speaking, vectors only describe *pure* quantum states: there are also *mixed* states, which need to be described differently; but we do not consider those here.

Let $\{|0\rangle, |1\rangle\}$ be an orthonormal basis for $\mathbb{C}^2$. We call this the *computational basis*. The induced basis on $\left(\mathbb{C}^2\right)^{\otimes n}$ is labelled by $\{|x\rangle\}_{x\in\{0,1\}^n}$ as a short-hand, e.g. we write $|00\ldots0\rangle$ instead of $|0\rangle\otimes|0\rangle\otimes\ldots\otimes|0\rangle$. This is exactly the same as the basis introduced in Section 2.1.

Holographic transformations also have a natural interpretation in quantum information theory: going from an n-qubit state $|f\rangle$ to $M^{\otimes n}|f\rangle$, where M is some invertible 2 by 2 matrix, is a 'stochastic local operation with classical communication' (SLOCC) [4, 15]. These are physical operations that can be applied locally (without needing access to more than one qubit at a time) using classical (i.e. non-quantum) communication between the sites where the different qubits are held, and which succeed with non-zero probability. Unlike holographic transformations, SLOCC operations do not need to be symmetric under interchange of the qubits: the most general SLOCC operation on an n-qubit state is given by $M_1\otimes M_2\otimes\ldots\otimes M_n$, where $M_1, M_2, \ldots M_n$ are invertible complex 2 by 2 matrices [15].

From now on, we will use standard Holant terminology (or notation) and quantum terminology (or notation) interchangeably, and sometimes mix the two.

4.1 Entanglement and its classification

One major difference between quantum theory and preceding theories of physics (known as 'classical physics') is the possibility of *entanglement* in states of multiple systems.

▶ **Definition 10.** A state of multiple systems is *entangled* if it cannot be written as a tensor product of states of individual systems.

Where a state involves more than two systems, it is possible for some of the systems to be entangled with each other and for other systems to be in a product state with respect to the former. We sometimes use the term *genuinely entangled state* to refer to a state in which no subsystem is in a product state with the others. The term *multipartite entanglement* refers to entangled states in which more than two qubits are mutually entangled. Non-degenerate signatures correspond to (not necessarily genuinely) entangled states.

Entanglement is an important resource in QC, where it has been shown that quantum speedups are impossible without the presence of unboundedly growing amounts of entanglement [22]. Similarly, it is a resource in QIT [28], featuring in protocols such as quantum teleportation [3] and quantum key distribution [16]. Many QIT protocols have the property that two quantum states can be used to perform the same task if one can be transformed into the other by SLOCC, motivating the following equivalence relation.

▶ **Definition 11.** Two n-qubit states are *equivalent under SLOCC* if one can be transformed into the other using SLOCC. More formally: suppose $|f\rangle$ and $|g\rangle$ are two n-qubit states. Then $|f\rangle \sim_{SLOCC} |g\rangle$ if and only if there exist invertible complex 2 by 2 matrices $M_1, M_2, \ldots, M_n$ such that $(M_1\otimes M_2\otimes\ldots\otimes M_n)|f\rangle = |g\rangle$.

The equivalence classes of this relation are called *entanglement classes* or *SLOCC classes*.

For two qubits, there is only one class of entangled states, i.e. all entangled two-qubit states are equivalent to $|00\rangle + |11\rangle$ under SLOCC. For three qubits, there are two classes of genuinely entangled states [15], called the GHZ class and the W class. The former contains states that are equivalent under SLOCC to the GHZ state $|\mathrm{GHZ}\rangle := |000\rangle + |111\rangle$, the latter those equivalent to the W state $|W\rangle := |001\rangle + |010\rangle + |100\rangle$. Given an arbitrary three-qubit state expressed in the computational basis, it is straightforward to determine its entanglement class [26]. For more than three qubits, there are infinitely many SLOCC classes. It is possible to partition these into families which share similar properties. Yet, so

far, there is no consensus on how to partition the classes: there are different schemes for partitioning even the four-qubit entanglement classes, yielding different families [31, 25, 2].

It is sometimes useful to generalise the definitions of GHZ and W states to n-qubit states. The generalised GHZ state on n qubits is $|\mathrm{GHZ}_n\rangle := |0\rangle^{\otimes n} + |1\rangle^{\otimes n}$, i.e. it is the state corresponding to the n-ary equality signature. The generalised W state on n qubits is defined as $|W_1\rangle := |1\rangle$ and $|W_n\rangle := |1\rangle \otimes |0\rangle^{\otimes n-1} + |0\rangle \otimes |W_{n-1}\rangle$ for $n > 1$, i.e. $|W_n\rangle$ corresponds to the n-ary indicator function for inputs of Hamming weight 1. We sometimes drop the word 'generalised' when talking about generalised GHZ or W states. It should be clear from context whether or not we mean the three-qubit state specifically.

4.2 The existing results in the quantum picture

Several of the existing dichotomies have straightforward descriptions in the quantum picture. The tractable cases of the Holant* dichotomy (cf. Section 3.1) can be described as follows:

- either there is no multipartite entanglement – this corresponds to the case $\mathcal{F} \subseteq \langle \mathcal{T} \rangle$, or
- there is GHZ-type multipartite entanglement but it is impossible to realise W-type multipartite entanglement – this corresponds to the cases $\mathcal{F} \subseteq \langle O \circ \mathcal{E} \rangle$ or $\mathcal{F} \subseteq \langle K \circ \mathcal{E} \rangle$, or
- there is W-type multipartite entanglement and it is impossible to realise GHZ-type multipartite entanglement – this corresponds to the case $\mathcal{F} \subseteq \langle K \circ \mathcal{M} \rangle$ or $\mathcal{F} \subseteq \langle KX \circ \mathcal{M} \rangle$.

By GHZ-type entanglement we mean states that are equivalent to generalised GHZ states under SLOCC, and similarly for W-type entanglement.

The tractable case of Holant^c (cf. Section 3.2) that does not appear in Holant* also has a natural description: in QIT, the states corresponding to affine signatures are known as *stabilizer states* [13]. These states and the associated operations play an important role in the context of quantum error-correcting codes [20] and are thus at the core of most attempts to build large-scale quantum computers [14]. The fragment of quantum theory consisting of stabilizer states and operations that preserve the set of stabilizer states can be efficiently simulated on a classical computer [20]; this result is known as the Gottesman-Knill theorem.

Thus, the Holant problem and QIT are linked not only by quantum algorithms being an inspiration for holographic ones: instead, the known tractable signature sets of various Holant problems correspond to state sets that are of independent interest in QC and QIT.

The restriction to algebraic numbers is not a problem from the quantum perspective, not even when considering the question of universal QC: there exist (approximately) universal sets of quantum operations where each operation can be described using algebraic complex coefficients. One such example is the Clifford+T gate set [5, 19].

5 Holant$^+$

Our new family of Holant problems, called Holant$^+$, sits in between Holant* and Holant^c. It has a small number of freely available signatures, which are all unary. Yet, using results from QIT, these can be shown to be sufficient for constructing the gadgets required to reduce to the dichotomies in Section 3.4. Formally:

$$\text{Holant}^+ (\mathcal{F}) = \text{Holant} (\mathcal{F} \cup \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}), \tag{7}$$

where $|+\rangle := |0\rangle + |1\rangle$ corresponds to the 'unary equality function' and $|-\rangle := |0\rangle - |1\rangle$ is a vector that is orthogonal to $|+\rangle$. In quantum theory, the set $\{|+\rangle, |-\rangle\}$ is known as the *Hadamard basis*, since they are related to the computational basis vectors by a Hadamard transformation (up to scalar factor): $\{|+\rangle, |-\rangle\} \doteq H \circ \{|0\rangle, |1\rangle\}$, where $H = \frac{1}{\sqrt{2}} \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$.

5.1 Why these free signatures?

The definition of $\#\text{HOLANT}^+$ is motivated by the following result from quantum theory.

▶ **Theorem 12** ([29],[18]). *Let $|\Psi\rangle$ be an n-system entangled state. For any two of the n systems, there exists a projection, onto a tensor product of states of the other $(n-2)$ systems, that leaves the two systems in an entangled state.*

Here, 'projection' means a (partial) inner product between $|\Psi\rangle$ and the tensor product of single-system states. The original proof of this statement in [29] was flawed but it was recently corrected [18]. The following corollary is not stated explicitly in either paper, but can be seen to hold by inspecting the proof in [18].

▶ **Corollary 13.** *Let $|\Psi\rangle$ be an n-qubit entangled state. For any two of the n qubits, there exists a projection of the other $(n-2)$ qubits onto a tensor product of computational and Hadamard basis states that leaves the two qubits in an entangled state.*

In other words, Theorem 12 holds when the systems are restricted to qubits and the projectors are restricted to products of computational and Hadamard basis states. Here, it is crucial to have projectors taken from two bases that are linked by the Hadamard transformation: the corollary works only in that case. We extend this result as follows.

▶ **Theorem 14.** *Let $|\Psi\rangle$ be an n-qubit entangled state with $n \geq 3$. There exists some choice of three of the n qubits and a projection of the other $(n-3)$ qubits onto a tensor product of computational and Hadamard basis states that leaves the three qubits in a genuinely entangled state.*

Proof (Sketch). If $n = 3$, $|\Psi\rangle$ itself is the desired state. For larger n, the theorem is proved inductively: we show that, given an n-qubit entangled state with $n > 3$, it is possible to project $(n-3)$ qubits in the desired way, assuming the same holds for all k-qubit genuinely entangled states with $3 \leq k \leq n$. The induction step is then proved by contradiction, employing the assumption that Theorem 14 does not hold for $(n+1)$-qubit states while Theorem 12 does. The full proof can be found in [1]. ◀

This result, which was not previously known in the QIT literature, is stronger than Theorem 12 in that we construct entangled three-qubit states rather than two-qubit ones. On the other hand, our result may not hold for all choices of three qubits: all we show is that there exists some choice of three qubits for which it does hold. The original proof of Theorem 14 in an earlier version of this paper was long and involved; this new shorter proof was suggested by Gachechiladze and Gühne [17].

5.2 The dichotomy theorem

Using Theorem 14, as well as Theorems 8 and 9, we prove our main result: a dichotomy for $\#\text{HOLANT}^+$ applying to complex, not necessarily symmetric signatures.

▶ **Theorem 15.** *Let $\mathcal{F}$ be a set of complex signatures. $\#\text{HOLANT}^+(\mathcal{F})$ is in FP if $\mathcal{F}$ satisfies one of the following conditions:*

- *$\#\text{HOLANT}^*(\mathcal{F})$ is in FP, or*
- *$\mathcal{F} \subseteq \mathcal{A}$.*

In all other cases, the problem is #P-hard.

The tractable cases are almost the same as those for symmetric Holant^c (Theorem 6), now without the symmetry restriction. The only difference is that the holographic transformations allowed in the affine case of the Holant^c dichotomy are trivial in the case of Holant$^+$: any transformation that maps $\{|=_2\rangle, |0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ to a subset of $\mathcal{A}$ must itself be in $\mathcal{A}$.

The tractability proof follows immediately by reduction to Holant* or #CSP, respectively. For the hardness proof, we use Theorem 14 to construct signatures corresponding to three-qubit entangled states. We then show that, unless we are in one of the tractable cases, it is possible to construct ternary gadgets with non-degenerate symmetric signatures. If the ternary symmetric signature is in the GHZ class, Theorem 8 applies. If the ternary symmetric signature is in the W class but not in $K \circ \mathcal{M}$ or $KX \circ \mathcal{M}$, we use Theorem 9. Finally, if the ternary symmetric signature is contained in $K \circ \mathcal{M}$, then by assumption the set of available signatures $\mathcal{F}$ must contain some signature that is not in $K \circ \mathcal{M}$ – otherwise, the problem is already known to be tractable. We show how to use such a signature to construct a binary symmetric signature that is not in $K \circ \mathcal{M}$. Then the desired result follows by Theorem 9. An analogous result holds with $KX \circ \mathcal{M}$ instead.

The gadget constructions for ternary symmetric signatures are given in Section 5.3. The gadget construction for a symmetric binary signature that is not in $K \circ \mathcal{M}$ (or $KX \circ \mathcal{M}$) follows in Section 5.4. Section 5.5 contains the hardness proof itself.

5.3 Symmetrising ternary signatures

The dichotomies given in Section 3.4 apply to symmetric ternary entangled signatures. The signatures constructed according to Theorem 14 are ternary and entangled, but they are not generally symmetric. Yet, these general ternary entangled signatures can be used to realise symmetric ones, possibly with the help of an additional binary non-degenerate signature. We prove this by distinguishing cases according to whether the ternary entangled signature constructed using Theorem 14 is in the GHZ or the W entanglement class.

First, consider a general GHZ-class state $|\psi\rangle$. By definition, there exist invertible complex 2 by 2 matrices A, B, C such that $|\psi\rangle = (A \otimes B \otimes C) |\mathrm{GHZ}\rangle$. We can draw the signature associated with $|\psi\rangle$ as the 'virtual gadget' shown in Figure 1a. The 'boxes' denoting the matrices are non-symmetric to indicate that A, B, C are not in general symmetric. The white dot represents the GHZ state. This notation is not meant to imply that the signatures A, B, C or the ternary equality signature are available on their own. Thinking of the signature as such a composite will simply make future arguments more straightforward. A similar argument can be applied if $|\psi\rangle$ is a W-class state, in which case the white dots in Figure 1 should be thought of as having signature $|W\rangle$.

In both cases, three vertices with the same ternary entangled signature can be connected to form the rotationally symmetric gadget shown in Figure 1b. In fact, the signature for that gadget is fully symmetric: its value depends only on the Hamming weight of the inputs. On the other hand, it may not be entangled or it may have the all-zero signature. For a general non-symmetric $|\psi\rangle$ there are three such symmetric gadgets that can be constructed by permuting the roles of A, B, and C in Figure 1b – in particular, which of the three ends up on the external edge of the gadget. This idea leads to the following lemmas.

▶ **Lemma 16.** *Let $|\psi\rangle$ be a three-qubit GHZ-class state, i.e. $|\psi\rangle = (A \otimes B \otimes C) |\mathrm{GHZ}\rangle$ for some invertible 2 by 2 matrices A, B, C. Then at least one of the three possible symmetric gadgets resulting from permutations of A, B, C in Figure 1b is non-degenerate unless $|\psi\rangle \in K \circ \mathcal{E}$ and is furthermore already symmetric.*

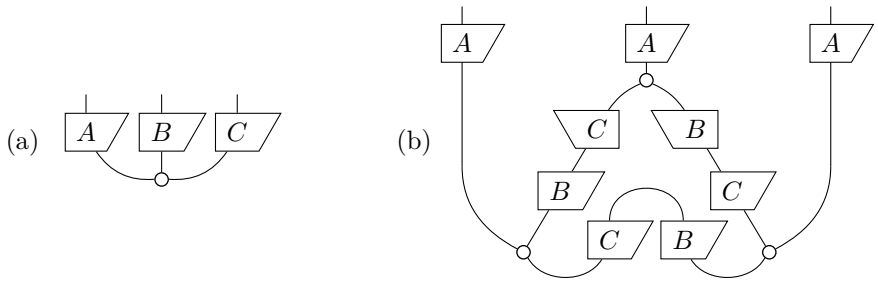


Figure 1 (a) A 'virtual gadget' for an entangled ternary signature based on the idea of SLOCC classes. (b) A symmetric gadget constructed from three copies of that ternary signature.

▶ **Lemma 17.** *Let $|\psi\rangle$ be a three-qubit W-class state, i.e. $|\psi\rangle = (A \otimes B \otimes C)|W\rangle$ for some invertible 2 by 2 matrices A, B, C. If $|\psi\rangle \in K \circ \mathcal{M}$ (or $|\psi\rangle \in KX \circ \mathcal{M}$), assume that we also have a two-qubit entangled state $|\phi\rangle$ that is not in $K \circ \mathcal{M}$ (or $KX \circ \mathcal{M}$, respectively). Then we can realise a symmetric three-qubit entangled state.*

5.4 Constructing binary signatures

We have shown in the previous section that it is possible to realise a non-degenerate ternary symmetric signature under some mild assumptions. Now, we show that if the full signature set $\mathcal{F}$ is not a subset of $K \circ \mathcal{M}$ (or $KX \circ \mathcal{M}$), it is possible to construct a symmetric binary gadget over $\mathcal{F} \cup \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ whose signature is not in $K \circ \mathcal{M}$ (or $KX \circ \mathcal{M}$, respectively). This signature can be used in Lemma 17, and a symmetric signature realised from it can also be used for a hardness proof according to Theorem 9.

▶ **Lemma 18.** *Suppose $|\psi\rangle$ is a genuinely entangled n-qubit state with $n \geq 2$, and $|\psi\rangle \notin K \circ \mathcal{M}$. Then there exists a non-degenerate binary gadget over $\{|\psi\rangle, |0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ with signature $|\varphi\rangle \notin K \circ \mathcal{M}$.*

The binary signature required in Lemma 17 is not required to be symmetric, only non-degenerate. The one in Theorem 9, on the other hand, does need to be symmetric.

▶ **Lemma 19.** *Suppose $|\psi\rangle \in K \circ \mathcal{M}$ is a three-qubit symmetric entangled state and $|\phi\rangle \notin K \circ \mathcal{M}$ is a two-qubit entangled state. Then there exists a gadget over $\{|\psi\rangle, |\phi\rangle, |0\rangle, |1\rangle, |\pm\rangle\}$ such that its signature $|\varphi\rangle$ is a two-qubit symmetric entangled state and $|\varphi\rangle \notin K \circ \mathcal{M}$.*

An analogous argument holds with KX instead of K. Hence, we can construct a non-degenerate symmetric binary signature satisfying the required properties whenever needed.

5.5 Sketch of the hardness proof

Suppose $\mathcal{F}$ is not in one of the tractable cases. Then, in particular, $\mathcal{F} \not\subseteq \langle \mathcal{T} \rangle$, i.e. $\mathcal{F}$ must contain multipartite entanglement (cf. Section 3.1). We can therefore use Theorem 14 to realise a ternary entangled signature. The quantum state associated with this signature must be in either the GHZ or the W SLOCC class.

In the GHZ case, either the state is already symmetric or it is possible to realise a non-degenerate symmetric ternary signature by Lemma 16. In the W case, if the ternary signature is not in $K \circ \mathcal{M}$ or $KX \circ \mathcal{M}$, it can be used to realise a non-degenerate ternary symmetric signature by Lemma 17. If the ternary signature is in $K \circ \mathcal{M}$, by Lemma 18, we can realise a binary signature that is not in $K \circ \mathcal{M}$ since by assumption $\mathcal{F} \not\subseteq K \circ \mathcal{M}$; and

similarly with KX instead of K. This then enables the use of Lemma 17. Hence if $\mathcal{F}$ is not one of the tractable sets, it is always possible to realise a non-degenerate symmetric ternary signature. Again, the quantum state associated with this signature must be in either the GHZ or the W SLOCC class.

If it is a GHZ class state, use the following lemma and corollary to reduce the problem to Theorem 8. This theorem yields #P-hardness unless $\mathcal{F}$ is a subset of $\langle O \circ \mathcal{E} \rangle$ or $\mathcal{A}$, which we assumed it was not.

▶ **Lemma 20.** *Let f be a signature and $\mathcal{G}$ a set of signatures. Then:*

$$\text{Holant}(\{f\} \cup \mathcal{G}) \equiv_T \text{Holant}(\{f, [1,0,1]\} \mid \mathcal{G} \cup \{[1,0,1]\}). \tag{8}$$

▶ **Corollary 21.** *Let f be a signature and $\mathcal{G}$ a set of signatures, and let M be an invertible 2 by 2 matrix. Then:*

$$\text{Holant}(\{M \circ f\} \cup \mathcal{G}) \equiv_T \text{Holant}\left(\left\{f, M^{-1} \circ [1,0,1]\right\} \,\middle|\, (\mathcal{G} \cup \{[1,0,1]\}) \circ M^T\right). \tag{9}$$

The corollary follows immediately from Lemma 20 and Theorem 2.

If the non-degenerate symmetric ternary signature $|\psi\rangle$ realised according to Section 5.3 is in the W class, then, by Theorem 9, the problem is #P-hard unless the signature is in $K \circ \mathcal{M}$ (or $KX \circ \mathcal{M}$). In the latter case, as by assumption $\mathcal{F} \not\subseteq K \circ \mathcal{M}$ (or $\mathcal{F} \not\subseteq KX \circ \mathcal{M}$), we can use Lemmas 18 and 19 to construct a symmetric binary signature $|\varphi\rangle$ that is not in $K \circ \mathcal{M}$ (or $KX \circ \mathcal{M}$, respectively).

Now, $\text{Holant}(\{|\varphi\rangle\} \mid \{|\psi\rangle\}) \leq_T \text{Holant}(\{|\varphi\rangle, |\psi\rangle\} \cup \mathcal{G})$ for any set $\mathcal{G}$. But if $|\psi\rangle \in K \circ \mathcal{M}$ and $|\varphi\rangle \notin K \circ \mathcal{M}$, then $\text{Holant}(\{|\varphi\rangle\} \mid \{|\psi\rangle\})$ is #P-hard by Theorem 9, and similarly with KX instead of K. Thus $\text{Holant}^+(\mathcal{F})$ is #P-hard whenever such $|\psi\rangle$ and $|\varphi\rangle$ are realisable over $\mathcal{F}$.

This concludes the investigation of all cases. We have therefore shown that Holant^+ is #P-hard in all but the listed cases. A full proof of this result can be found in [1].

6 Conclusions

Applying knowledge from QIT to Holant problems, we find that several tractable classes of existing dichotomies have concise descriptions in the framework of quantum entanglement. Motivated by this and by existing results in entanglement theory, we define a new Holant family, Holant^+, fitting between the known families Holant^* and Holant^c. We derive a full dichotomy for this family, which is closely related to the dichotomy for symmetric Holant^c [10]. It may therefore be a useful stepping stone towards a full Holant^c dichotomy, and thus to a full dichotomy for all Holant problems.

We also prove a new result in entanglement theory: given any n-qubit genuinely entangled state, it is possible to find some subset of $(n-3)$ qubits and a projector which is a tensor product of $(n-3)$ computational and Hadamard basis states such that the projection leaves the remaining three qubits in a genuinely entangled state. This is a generalisation of a similar result about constructing two-qubit entangled states [29, 18], though our result is slightly weaker in some aspects, which it may be possible to strengthen in future work.

We expect that further analysis of Holant problems using methods from QIT and QC will lead to further new insights, both into the complexity of Holant problems and into entanglement or other areas of quantum theory.

Acknowledgements. I would like to thank Ashley Montanaro, both for introducing me to Holant problems and for helpful comments on earlier drafts of this paper. I would also like to thank Mariami Gachechiladze and Otfried Gühne for pointing out a significantly shorter and more elegant proof of Theorem 14, and for letting me use it here.

References

1 Miriam Backens. A new Holant dichotomy inspired by quantum computation. *arXiv:1702.00767 [quant-ph]*, February 2017. Full version. URL: `http://arxiv.org/abs/1702.00767`.

2 Miriam Backens. Number of superclasses of four-qubit entangled states under the inductive entanglement classification. *Physical Review A*, 95(2):022329, February 2017. `doi:10.1103/PhysRevA.95.022329`.

3 Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, 70(13):1895–1899, March 1993. `doi:10.1103/PhysRevLett.70.1895`.

4 Charles H. Bennett, Sandu Popescu, Daniel Rohrlich, John A. Smolin, and Ashish V. Thapliyal. Exact and asymptotic measures of multipartite pure-state entanglement. *Physical Review A*, 63(1):012307, December 2000. `doi:10.1103/PhysRevA.63.012307`.

5 P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. On universal and fault-tolerant quantum computing: A novel basis and a new constructive proof of universality for Shor's basis. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 486–494, New York, October 1999. IEEE. `doi:10.1109/SFFCS.1999.814621`.

6 J. Cai, P. Lu, and M. Xia. Dichotomy for Holant* Problems of Boolean Domain. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, Proceedings, pages 1714–1728. Society for Industrial and Applied Mathematics, January 2011. `doi:10.1137/1.9781611973082.132`.

7 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Graph Homomorphisms with Complex Values: A Dichotomy Theorem. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, number 6198 in Lecture Notes in Computer Science, pages 275–286. Springer Berlin Heidelberg, July 2010. Full version at arXiv:0903.4728. `doi:10.1007/978-3-642-14165-2_24`.

8 Jin-Yi Cai and Vinay Choudhary. Valiant's Holant Theorem and Matchgate Tensors. In Jin-Yi Cai, S. Barry Cooper, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, number 3959 in Lecture Notes in Computer Science, pages 248–261. Springer Berlin Heidelberg, May 2006. `doi:10.1007/11750321_24`.

9 Jin-Yi Cai, Heng Guo, and Tyson Williams. A Complete Dichotomy Rises from the Capture of Vanishing Signatures: Extended Abstract. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC'13, pages 635–644, New York, NY, USA, 2013. ACM. `doi:10.1145/2488608.2488687`.

10 Jin-Yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and Back: Dichotomy for Holant^c Problems. *Algorithmica*, 64(3):511–533, March 2012. `doi:10.1007/s00453-012-9626-6`.

11 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holant Problems and Counting CSP. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC'09, pages 715–724, New York, NY, USA, 2009. ACM. `doi:10.1145/1536414.1536511`.

12 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted Boolean #CSP. *Journal of Computer and System Sciences*, 80(1):217–236, February 2014. `doi:10.1016/j.jcss.2013.07.003`.

13 Jeroen Dehaene and Bart De Moor. Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Physical Review A*, 68(4):042318, October 2003. doi:10.1103/PhysRevA.68.042318.

14 Simon J Devitt, William J Munro, and Kae Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7):076001, July 2013. doi:10.1088/0034-4885/76/7/076001.

15 W. Dür, G. Vidal, and J. I. Cirac. Three qubits can be entangled in two inequivalent ways. *Physical Review A*, 62(6):062314, November 2000. doi:10.1103/PhysRevA.62.062314.

16 Artur K. Ekert. Quantum cryptography based on Bell's theorem. *Physical Review Letters*, 67(6):661–663, August 1991. doi:10.1103/PhysRevLett.67.661.

17 Mariami Gachechiladze and Otfried Gühne, February 2017. Personal communication.

18 Mariami Gachechiladze and Otfried Gühne. Completing the proof of "Generic quantum nonlocality". *Physics Letters A*, 381(15):1281–1285, April 2017. doi:10.1016/j.physleta.2016.10.001.

19 Brett Giles and Peter Selinger. Exact synthesis of multiqubit Clifford+T circuits. *Physical Review A*, 87(3):032332, March 2013. doi:10.1103/PhysRevA.87.032332.

20 Daniel Gottesman. The Heisenberg Representation of Quantum Computers. *arXiv:quant-ph/9807006*, July 1998. Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics, eds. S. P. Corney, R. Delbourgo, and P. D. Jarvis, pp. 32–43 (Cambridge, MA, International Press, 1999). URL: http://arxiv.org/abs/quant-ph/9807006.

21 Sangxia Huang and Pinyan Lu. A Dichotomy for Real Weighted Holant Problems. *computational complexity*, 25(1):255–304, March 2016. doi:10.1007/s00037-015-0118-3.

22 Richard Jozsa and Noah Linden. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032, 2003. doi:10.1098/rspa.2002.1097.

23 Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *J. ACM*, 22(1):155–171, January 1975. doi:10.1145/321864.321877.

24 L. Lamata, J. León, D. Salgado, and E. Solano. Inductive classification of multipartite entanglement under stochastic local operations and classical communication. *Physical Review A*, 74(5):052336, November 2006. doi:10.1103/PhysRevA.74.052336.

25 L. Lamata, J. León, D. Salgado, and E. Solano. Inductive entanglement classification of four qubits under stochastic local operations and classical communication. *Physical Review A*, 75(2):022318, February 2007. doi:10.1103/PhysRevA.75.022318.

26 Dafa Li, Xiangrong Li, Hongtao Huang, and Xinxin Li. Simple criteria for the SLOCC classification. *Physics Letters A*, 359(5):428–437, December 2006. doi:10.1016/j.physleta.2006.07.004.

27 Jiabao Lin and Hanpin Wang. The Complexity of Holant Problems over Boolean Domain with Non-negative Weights. *arXiv: 1611.00975 [cs]*, November 2016. URL: http://arxiv.org/abs/1611.00975.

28 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2010.

29 Sandu Popescu and Daniel Rohrlich. Generic quantum nonlocality. *Physics Letters A*, 166(5–6):293–297, June 1992. doi:10.1016/0375-9601(92)90711-T.

30 L. Valiant. Holographic Algorithms. *SIAM Journal on Computing*, 37(5):1565–1594, January 2008. doi:10.1137/070682575.

31 F. Verstraete, J. Dehaene, B. De Moor, and H. Verschelde. Four qubits can be entangled in nine different ways. *Physical Review A*, 65(5):052112, April 2002. doi:10.1103/PhysRevA.65.052112.

Efficient Quantum Algorithms for Simulating Lindblad Evolution*†

Richard Cleve[1] and Chunhao Wang[2]

1 **Institute for Quantum Computing, University of Waterloo, Waterloo, Canada; and**
Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada; and
Canadian Institute for Advanced Research, Toronto, Canada
cleve@uwaterloo.ca

2 **Institute for Quantum Computing, University of Waterloo, Waterloo, Canada; and**
Cheriton School of Computer Science, University of Waterloo, Canada
c265wang@uwaterloo.ca

Abstract

We consider the natural generalization of the Schrödinger equation to *Markovian open system dynamics*: the so-called the Lindblad equation. We give a quantum algorithm for simulating the evolution of an n-qubit system for time t within precision ϵ. If the Lindbladian consists of $\text{poly}(n)$ operators that can each be expressed as a linear combination of $\text{poly}(n)$ tensor products of Pauli operators then the gate cost of our algorithm is $O(t\,\text{polylog}(t/\epsilon)\text{poly}(n))$. We also obtain similar bounds for the cases where the Lindbladian consists of local operators, and where the Lindbladian consists of sparse operators. This is remarkable in light of evidence that we provide indicating that the above efficiency is impossible to attain by first expressing Lindblad evolution as Schrödinger evolution on a larger system and tracing out the ancillary system: the cost of such a *reduction* incurs an efficiency overhead of $O(t^2/\epsilon)$ even before the Hamiltonian evolution simulation begins. Instead, the approach of our algorithm is to use a novel variation of the "linear combinations of unitaries" construction that pertains to channels.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Quantum algorithms, open quantum systems, Lindblad simulation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.17

1 Introduction

The problem of simulating the evolution of closed systems (captured by the Schrödinger equation) was proposed by Feynman [12] in 1982 as a motivation for building quantum computers. Since then, several quantum algorithms have appeared for this problem (see section 1.1 for references to these algorithms). However, many quantum systems of interest are not closed but are well-captured by the Lindblad Master equation [21, 13]. Examples exist in quantum physics [20, 32], quantum chemistry [25, 27], and quantum biology [11, 14, 26]. Lindblad evolution also arises in quantum computing and quantum information in the

* The full version of this paper is available at http://arxiv.org/abs/1612.09512.

† This research was supported in part by Canada's NSERC and an NSERC Canada Graduate Scholarship (Doctoral).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 17; pp. 17:1–17:14

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

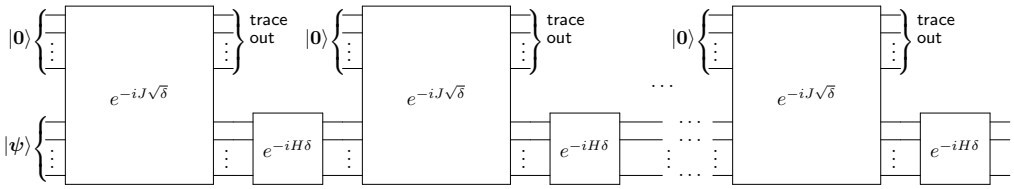


Figure 1 Lindblad evolution for time t approximated by unitary operations. There are N iterations and $\delta = t/N$. This converges to Lindblad evolution as $N \to \infty$.

context of entanglement preparation [19, 16, 29], thermal state preparation [15], quantum state engineering [31], and studying the noise of quantum circuits [24].

We consider the computational cost of simulating the evolution of an n-qubit quantum state for time t under the Lindblad Master equation

$$\dot{\rho} = -i[H, \rho] + \sum_{j=1}^{m} \Big(L_j \rho L_j^\dagger - \frac{1}{2} L_j^\dagger L_j \rho - \frac{1}{2} \rho L_j^\dagger L_j \Big), \tag{1}$$

(representing Markovian open system dynamics), where H is a Hamiltonian and $L_1, \ldots, L_m$ are linear operators. By *simulate the evolution*, we mean: provide a quantum circuit that computes the quantum channel corresponding to evolution by Eq. (1) for time t within precision ϵ. The quantum circuit must be independent of the input state, which is presumed to be unknown. When $L_1 = \cdots = L_m = 0$, Eq. (1) is the Schrödinger equation.

Eq. (1) can be viewed as an idealization of the frequently occurring physical scenario where a quantum system evolves jointly with a large external environment in a manner where information dissipates from the system into the environment. In quantum information theoretic terms, Lindblad evolution is a continuous-time process that, for any evolution time, is a quantum channel. Moreover, Lindblad evolution is *Markovian* in the sense that, for any $\delta > 0$, the state at time $t + \delta$ is a function of the state at time t alone (i.e., is independent of the state before time t).

Lindblad evolution can be intuitively thought of as Hamiltonian evolution in a larger system that includes an ancilla register, but where the ancilla register is being continually reset to its initial state. To make this more precise, consider a time interval $[0, t]$, and divide it into N subintervals of length $\frac{t}{N}$ each. At the beginning of each subinterval, reset the state of the ancilla register to its initial state, and then let the joint system-ancilla evolve under a Hamiltonian J and the system itself evolve under H. Let the evolution time for J be $\sqrt{t/N}$ and the evolution time for H be t/N. This process, illustrated in Fig. 1, converges to true Lindblad evolution as N approaches ∞.

For the specific evolution described by Eq. (1), it suffices to set the ancilla register to $\mathbb{C}^{m+1}$ and the Hamiltonian J to the block matrix

$$J = \begin{pmatrix} 0 & L_1^\dagger & \cdots & L_m^\dagger \\ L_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_m & 0 & \cdots & 0 \end{pmatrix}. \tag{2}$$

A remarkable property of this way of representing Lindblad evolution is that the rate at which the Hamiltonian J evolves is effectively infinite: Lindblad evolution for time t/N is simulated by a process that includes evolution by J for time $\sqrt{t/N}$, so the rate of the

evolution scales as

$$\frac{\sqrt{t/N}}{t/N} = \sqrt{\frac{N}{t}}, \tag{3}$$

which diverges as $N \to \infty$. Moreover, the total Hamiltonian evolution time of J in Fig. 1 is $N\sqrt{t/N} = \sqrt{Nt}$, which also diverges. In the Appendix A we prove that, in general, the above scaling phenomenon is necessary for simulating time-independent Lindblad evolution in terms of time-independent Hamiltonian evolution along the lines of the overall structure of Fig. 1. In this sense, *exact* Lindblad evolution for finite time does not directly correspond to Hamiltonian evolution for *any* finite time. On the other hand, it can be shown that if the scaling of N is at least t^3/ϵ^2 then the final state is an *approximation* within ϵ. Note that then the corresponding total evolution time for J scales as $\sqrt{(t^3/\epsilon^2)t} = t^2/\epsilon$. Therefore, quantum algorithms that simulate Lindblad evolution by first applying the above reduction to Hamiltonian evolution and then efficiently simulating the Hamiltonian evolution are likely to incur scaling that is at least t^2/ϵ.

Here we are interested in whether much more efficient simulations of Lindblad evolution are possible, such as $O(t \operatorname{polylog}(t/\epsilon))$.

1.1 Previous work

Simulating Hamiltonian evolution. Hamiltonian evolution (a.k.a. Schrödinger evolution) is the special case of Eq. (1) where $L_j = 0$ for all j. This simulation problem has received considerable attention since Feynman [12] proposed this as a motivation for building quantum computers; see for example [22, 1, 8, 2, 3, 5, 4, 18, 23, 28, 6]. Some of the recent methods obtain a scaling that is $O(t \operatorname{polylog}(t/\epsilon)\operatorname{poly}(n))$, thereby exceeding what can be accomplished by the longstanding Trotter-Suzuki methods [30].

Simulating Lindblad evolution. The natural generalization from closed systems to Markovian open systems in terms of the Lindblad equation has received much less attention. Kliesch *et al.* [17] give a quantum algorithm for simulating Lindblad evolution in the case where each of $H, L_1, \ldots, L_m$ can be expressed as a sum of local operators (i.e., which act on a constant number of qubits). The cost of this algorithm with respect to t and ϵ (omitting factors of $\operatorname{poly}(n)$) is $O(t^2/\epsilon)$. Childs and Li [9] improve this to $O(t^{1.5}/\sqrt{\epsilon})$ and also give an $O((t^2/\epsilon)\operatorname{polylog}(t/\epsilon))$ query algorithm for the case where the operators in Eq. (1) are sparse and represented in terms of an oracle. Another result in [9] is an $\Omega(t)$ lower bound for the query complexity for time t when Eq. (1) has $H = 0$ and $m = 1$.

As far as we know, none of the previous algorithms for simulating Lindblad evolution has cost $O(t \operatorname{polylog}(t/\epsilon)\operatorname{poly}(n))$, which is the performance that we attain. Our results are summarized precisely in the next subsection (subsection 1.2).

We note that there are simulation algorithms that solve problems that are related to but different from ours, such as [7], which does not produce the final state; rather it simulates the expectation of an observable applied to the final state. We do not know how to adapt these techniques to produce the unmeasured final state instead.

Finally, we note that there are interesting classical algorithmic techniques for simulating Lindblad evolution that are feasible when the dimension of the Hilbert space (which is 2^n, for n qubits) is not too large—but these do not carry over to the context of quantum algorithms (where n can be large). In the classical setting, since the state is known (and stored) explicitly, various "unravellings" of the process that are state-dependent can be simulated. For example, the random variable corresponding to "the next jump time" (which is highly state-dependent)

can be simulated. In the context of quantum algorithms, the input state is unknown and cannot be measured without affecting it.

1.2 New results

Eq. (1) can be written as $\dot{\rho} = \mathcal{L}[\rho]$, where $\mathcal{L}$ is a *Lindbladian*, defined as a mapping of the form

$$\mathcal{L}[\rho] = -i[H,\ \rho] + \sum_{j=1}^{m}\Big(L_j \rho L_j^\dagger - \frac{1}{2}L_j^\dagger L_j \rho - \frac{1}{2}\rho L_j^\dagger L_j\Big), \tag{4}$$

for operators $H, L_1, \ldots, L_m$ on the Hilbert space $\mathcal{H} = \mathbb{C}^{2^n}$ (n qubits) with H Hermitian. Evolution under Eq. (1) for time t corresponds to the quantum map $e^{\mathcal{L}t}$ (which is a channel for any $t \geq 0$).

Each of the operators $H, L_1, \ldots, L_m$ corresponds to a $2^n \times 2^n$ matrix. The simulation algorithm is based on a succinct specification of these matrices. Our succinct specification is as a *linear combination of q Paulis*, defined as

$$H = \sum_{k=0}^{q-1} \beta_{0k} V_{0k} \tag{5}$$

$$L_j = \sum_{k=0}^{q-1} \beta_{jk} V_{jk}, \tag{6}$$

where, for each $j \in \{0, \ldots, m\}$ and $k \in \{0, \ldots, q-1\}$, V_{jk} is an n-fold tensor product of Paulis (I, σ_x, σ_y, σ_z) and a scalar phase $e^{i\theta}$ ($\theta \in [0, 2\pi]$), and $\beta_{jk} \geq 0$.

In the evolution $e^{\mathcal{L}t}$, it is possible to scale up $\mathcal{L}$ by some factor while reducing t by the same factor, i.e., $e^{\mathcal{L}t}[\rho] = e^{(c\mathcal{L})\frac{t}{c}}[\rho]$ for any $c > 0$[1]. This reduces the simulation time but transfers the cost into the magnitude of $\mathcal{L}$. To normalize this cost, we define a norm based on the specification of $\mathcal{L}$.

Define the norm[2] of a specification of a Lindbladian $\mathcal{L}$ as a linear product of Paulis as

$$\|\mathcal{L}\|_{\mathsf{pauli}} = \sum_{k=0}^{q-1} \beta_{0k} + \sum_{j=1}^{m}\Big(\sum_{k=0}^{q-1} \beta_{jk}\Big)^2. \tag{7}$$

Our main result is the following theorem.

▶ **Theorem 1.** *Let $\mathcal{L}$ be a Lindbladian presented as a linear combination of q Paulis. Then, for any $t > 0$ and $\epsilon > 0$, there exists a quantum circuit of size*

$$O\left(m^2 q^2 \tau \frac{(\log(mq\tau/\epsilon) + n)\log(\tau/\epsilon)}{\log\log(\tau/\epsilon)}\right) \tag{8}$$

that implements a quantum channel $\mathcal{N}$, such that $\left\|\mathcal{N} - e^{\mathcal{L}t}\right\|_\diamond \leq \epsilon$, where $\tau = t\,\|\mathcal{L}\|_{\mathsf{pauli}}$.

---

[1] $c\mathcal{L}$ denotes the mapping obtained from $\mathcal{L}$ with H multiplied by c and each L_j multiplied by $\sqrt{c}$.

[2] For simplicity we use the terminology $\|\mathcal{L}\|_{\mathsf{pauli}}$ even though the quantity is not directly a function of the mapping $\mathcal{L}$. However, $\|c\mathcal{L}\|_{\mathsf{pauli}} = c\|\mathcal{L}\|_{\mathsf{pauli}}$ if $c\mathcal{L}$ denotes the expression in Eq. (4) with the factor c multiplied through.

Remarks

1. The proof of Theorem 1 is sketched in section 4 and is shown in the full version of this paper [10]. A main novel ingredient of the proof is Lemma 3, concerning a variant of the "linear combination of unitaries" construction that is suitable for channels (explained in sections 2 and 3).
2. The factor $\|\mathcal{L}\|_{\mathsf{pauli}}$ corresponding to the coefficients of the specification as a linear combination of Paulis is a natural generalization to the case of Lindbladians of a similar factor for Hamiltonians that appears in [3].
3. When $m, q \in \mathrm{poly}(n)$, the gate complexity in Theorem 1 simplifies to

$$O\left(\tau \frac{\log(\tau/\epsilon)^2}{\log\log(\tau/\epsilon)} \mathrm{poly}(n)\right). \tag{9}$$

4. A Lindbladian $\mathcal{L}$ is *local* if

$$H = \sum_{j=1}^{m'} H_j, \tag{10}$$

where $H_1, \ldots, H_{m'}$ and also $L_1, \ldots, L_m$ are local (i.e., they each act on a constant number of qubits). A *local* specification of $\mathcal{L}$ is as $H_1, \ldots, H_{m'}, L_1, \ldots, L_m$ and we define its norm as

$$\|\mathcal{L}\|_{\mathsf{local}} = \sum_{j=1}^{m'} \|H_j\| + \sum_{j=1}^{m} \|L_j\|^2. \tag{11}$$

For local Lindbladians, Theorem 1 reduces to the following.

▶ **Corollary 2.** *If $\mathcal{L}$ is a local Lindbladian then the gate complexity for simulating $e^{\mathcal{L}t}$ with precision ϵ is*

$$O\left((m+m')\,\tau \frac{\log((m+m')\tau/\epsilon)\log(\tau/\epsilon)}{\log\log(\tau/\epsilon)}\right), \tag{12}$$

where $\tau = t\,\|\mathcal{L}\|_{\mathsf{local}}$.

5. We also consider *sparse* Lindbladians (see [9] for various definitions, extending definitions and specifications of sparse Hamiltonians [1]). Here, we define a Lindbladian to have d-sparse operators if $H, L_1, \ldots, L_m$ each have at most d non-zero entries in each row/column. A *sparse specification* of such a Lindbladian $\mathcal{L}$ is as a black-box that provides the positions and values of the non-zero entries of each row/column of $H, L_1, \ldots, L_m$ via queries. Define the norm of any specification of a Lindbladian in terms of operators $H, L_1, \ldots, L_m$ as

$$\|\mathcal{L}\|_{\mathsf{ops}} = \|H\| + \sum_{j=1}^{m} \|L_j\|^2. \tag{13}$$

The query complexity and gate complexity for simulating d-sparse Lindbladians $\mathcal{L}$ are

$$O\big(\tau\,\mathrm{polylog}(mq\tau/\epsilon)\mathrm{poly}(d,n)\big), \tag{14}$$

where $\tau = t\|\mathcal{L}\|_{\mathsf{ops}}$. We sketch the analysis in the full version of this paper [10].
6. We expect some of the methodologies in [3, 4, 23, 28] to be adaptable to the Lindblad evolution simulation problem (in conjunction with our variant of the LCU construction and oblivious amplitude amplification), but have not investigated this.

2 Brief summary of novel techniques

As noted in subsection 1.1, for the case of Hamiltonian evolution, a series of recent quantum algorithms whose scaling is $O(t\,\mathrm{polylog}(t/\epsilon))$ have been discovered which improve on what has been accomplished using the longstanding Trotter-Suzuki decomposition. One of the main tools that these algorithms employ is a remarkable circuit construction that is based on a certain decomposition of *unitary* operations (or *near-unitary* operations) into a linear combination of unitaries. We refer to this construction as the *standard LCU method*.

For the case of Lindblad evolution, the operations that arise are *channels* that are not generally unitary. Some channels are *mixed unitary*, which means that they can be expressed as a randomly chosen unitary (say with probabilities $p_0, \ldots, p_{m-1}$ on the unitaries $U_0, \ldots, U_{m-1}$). For such channels, the standard LCU method can be adapted along the lines of first randomly sampling $j \in \{0, \ldots, m-1\}$ and then applying the standard LCU method to the unitary U_j. However, there exist channels that are *not* mixed unitary—and such channels can arise from the Lindblad equation. A different reductionist approach is to express these channels in the Stinespring form, as unitary operations that act on a larger system, and then apply the standard LCU method to *those* unitaries; however, as we explain in subsection 2.1, this approach performs poorly. We take a different approach that does not involve a reduction to the unitary case: we have developed a new variant of the LCU method that is for channels. This is explained in section 3.

Another new technique that we employ is an Oblivious Amplitude Amplification algorithm for isometries (as opposed to unitaries), which is noteworthy because a reductionist approach based on extending isometries to unitaries does not work. Roughly speaking, this is because our LCU construction turns out to produce an isometry (corresponding to a purification of the channel); however, it does not produce a unitary extension of that isometry.

2.1 The standard LCU method performs poorly on Stinespring dilations

Here we show in some technical detail why the standard LCU method performs poorly for Stinespring dilations of channels. The standard LCU method (explained in detail in Sec. 2.1 of [18]) for a unitary V expressible as a linear combination of unitaries as $V = \alpha_0 U_0 + \cdots + \alpha_{m-1} U_{m-1}$ is a circuit construction W that has the property

$$W|0\rangle|\psi\rangle = \sqrt{p}|0\rangle V|\psi\rangle + \sqrt{1-p}|\Phi^{\perp}\rangle \tag{15}$$

where $|\Phi^{\perp}\rangle$ has zero amplitude in states with first register $|0\rangle$ (i.e., $(|0\rangle\langle 0| \otimes I)|\Phi^{\perp}\rangle = 0$) and

$$p = \frac{1}{(\sum_{j=0}^{m-1} \alpha_j)^2} \tag{16}$$

is the success probability (that arises if the first indicator register is measured).

Consider the *amplitude damping channel*, which has two Kraus operators with the following LCU decompositions

$$A_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\delta} \end{bmatrix} = \alpha_{00} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \alpha_{01} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & \sqrt{\delta} \\ 0 & 0 \end{bmatrix} = \alpha_{10} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \alpha_{11} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix},$$

where $\alpha_{00} = \frac{1+\sqrt{1-\delta}}{2}$, $\alpha_{01} = \frac{1-\sqrt{1-\delta}}{2}$, $\alpha_{10} = \frac{\sqrt{\delta}}{2}$, $\alpha_{11} = \frac{\sqrt{\delta}}{2}$. Evolving an amplitude damping process for time t yields this channel with $\delta = 1 - e^{-t}$. When $t \ll 1$, $\delta \approx t$, $\alpha_{00} \approx 1 - t/4$, and $\alpha_{01} \approx t/4$.

A Stinespring dilation of V and its LCU decomposition can be derived from the above LCU decompositions of A_0 and A_1 as

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{1-\delta} & -\sqrt{\delta} & 0 \\ 0 & \sqrt{\delta} & \sqrt{1-\delta} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \alpha_{00} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \alpha_{01} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$+ \alpha_{10} \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} + \alpha_{11} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}.$$

Applying the standard LCU method here results in a success probability (computed from Eq. (16)) of

$$\frac{1}{\left(\alpha_{00}+\alpha_{01}+\alpha_{10}+\alpha_{11}\right)^2} = \frac{1}{\left(1+\sqrt{\delta}\right)^2} = 1 - 2\sqrt{\delta} + \Theta(\delta).$$

For small time evolution t, the failure probability is $\Theta(\sqrt{t})$, which is prohibitively expensive. It means that the process can be repeated at most $\Theta(1/\sqrt{t})$ times until the cumulative failure probability becomes a constant. The amount of evolution time (of the amplitude damping process) that this corresponds to is

$$\Theta\Big(\frac{1}{\sqrt{t}}\Big) \cdot t = \Theta(\sqrt{t}),$$

which is subconstant as $t \to 0$. This creates a problem in the general Lindblad simulation.

Our new LCU method for channels (explained in section 3) achieves the higher success probability

$$\frac{1}{\left(\alpha_{00}+\alpha_{01}\right)^2 + \left(\alpha_{10}+\alpha_{11}\right)^2} = \frac{1}{1+\delta} = 1 - \delta + \Theta(\delta^2).$$

For small time evolution t, the failure probability is $\Theta(t)$. Now, the process can be repeated $\Theta(1/t)$ times until the cumulative failure probability becomes a constant, which corresponds to evolution time

$$\Theta\Big(\frac{1}{t}\Big) \cdot t = \Theta(1),$$

which is constant as $t \to 0$. Since this is consistent with what arises in the algorithm of simulating Hamiltonian evolution in [2, 3], the methodologies used therein, with various adjustments, can be used to obtain the simulation bounds.

3 New LCU method for channels and completely positive maps

Let $A_0, \ldots, A_{m-1}$, linear operators on $\mathbb{C}^{2^n}$ (n-qubit states), be the Kraus operators of a channel. Suppose that, for each $j \in \{0, \ldots, m-1\}$, we have a decomposition of A_j as a linear combination of unitaries in the form

$$A_j = \sum_{k=0}^{q-1} \alpha_{jk} U_{jk}, \tag{17}$$

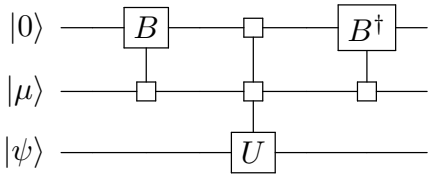


Figure 2 The circuit W for simulating a channel using the new LCU method.

where, for each $j \in \{0, \ldots, m-1\}$ and $k \in \{0, \ldots, q-1\}$, $\alpha_{jk} \geq 0$ and U_{jk} is unitary.

The objective is to implement the channel in terms of the implementations of U_{jk}'s. We will describe a circuit W and fixed state $|\mu\rangle$ such that, for any n-qubit state $|\psi\rangle$,

$$W|0\rangle|\mu\rangle|\psi\rangle = \sqrt{p}|0\rangle \left(\sum_{j=0}^{m-1} |j\rangle A_j|\psi\rangle \right) + \sqrt{1-p}|\Phi^{\perp}\rangle, \tag{18}$$

where $(|0\rangle\langle 0| \otimes I \otimes I)|\Phi^{\perp}\rangle = 0$ and

$$p = \frac{1}{\sum_{j=0}^{m-1} (\sum_{k=0}^{q-1} \alpha_{jk})^2} \tag{19}$$

is called the success probability parameter (which is realized if the first register is measured). Note that the isometry $|\psi\rangle \mapsto \sum_{j=0}^{m-1} |j\rangle A_j|\psi\rangle$ is the channel in purified form.

The circuit W is in terms of two gates. One gate is a *multiplexed-U gate*, denoted by multi-U such that, for all $j \in \{0, \ldots, m-1\}$ and $k \in \{0, \ldots, q-1\}$,

$$\text{multi-}U|k\rangle|j\rangle|\psi\rangle = |k\rangle|j\rangle U_{jk}|\psi\rangle. \tag{20}$$

The other gate is a *multiplexed-B gate*, denoted by multi-B, such that, for all $j \in \{0, \ldots, m-1\}$,

$$\text{multi-}B|0\rangle|j\rangle = \left(\frac{1}{\sqrt{s_j}} \sum_{k=0}^{q-1} \sqrt{\alpha_{jk}}|k\rangle \right) |j\rangle, \tag{21}$$

where

$$s_j = \sum_{k=0}^{q-1} \alpha_{jk}. \tag{22}$$

Define the state $|\mu\rangle$ (in terms of $s_0, \ldots, s_{m-1}$ from Eq. (22))

$$|\mu\rangle = \frac{1}{\sqrt{\sum_{j=1}^{m-1} s_j^2}} \sum_{j=0}^{m-1} s_j|j\rangle. \tag{23}$$

Define the circuit W (acting on $\mathbb{C}^q \otimes \mathbb{C}^m \otimes \mathbb{C}^{2^n}$) as

$$W = (\text{multi-}B^{\dagger} \otimes I)\text{multi-}U(\text{multi-}B \otimes I). \tag{24}$$

The LCU construction with the circuit W with its initial state $|0\rangle \otimes |\mu\rangle \otimes |\psi\rangle$ is illustrated in Fig. 2.

In this figure, we refer to the first register as the *indicator register* (as it indicates whether the computation succeeds at the end of this operation), the second register as the *purifier*

register (as it is used to purify the channel when the computation succeeds), and the third register as the *system register* (as it contains the state being evolved).

In the following lemma, Eq. (18) is shown to apply where $A_0, \ldots, A_{m-1}$ are arbitrary linear operators (i.e., Kraus operators of a completely positive map that is not necessarily trace preserving). If the map is also trace preserving then $\sum_{j=0}^{m-1} |j\rangle A_j|\psi\rangle$ and $|\Phi^{\perp}\rangle$ are normalized states and the success probability parameter p is the actual success probability realized if the first register is measured; otherwise, these need not be the case. In subsequent sections, we will apply this lemma in a context where the trace preserving condition is *approximately* satisfied.

▶ **Lemma 3.** *Let $A_0, \ldots, A_{m-1}$ be the Kraus operators of a completely positive map. Suppose that each A_j can be written in the form of Eq.* (17)*. Let multi-U, multi-B, W, and $|\mu\rangle$ be defined as above. Then applying the unitary operator W on any state of the form $|0\rangle|\mu\rangle|\psi\rangle$ produces the state*

$$\sqrt{p}|0\rangle \left(\sum_{j=0}^{m-1} |j\rangle A_j|\psi\rangle \right) + \sqrt{1-p}|\Phi^{\perp}\rangle,$$

where $(|0\rangle\langle 0| \otimes I \otimes I)|\Phi^{\perp}\rangle = 0$, and

$$p = \frac{1}{\sum_{j=0}^{m-1} \left(\sum_{k=0}^{q-1} \alpha_{jk} \right)^2}.$$

Proof. First consider the state $|0\rangle|j\rangle|\psi\rangle$ for any $j \in \{0, \ldots, m-1\}$. Applying W on this state is the standard LCU method [18]:

$$\begin{aligned}
W|0\rangle|j\rangle|\psi\rangle =& (\text{multi-}B^{\dagger} \otimes I)\text{multi-}U(\text{multi-}B \otimes I)|0\rangle|j\rangle|\psi\rangle && (25)\\
=& \frac{1}{\sqrt{s_j}}(\text{multi-}B^{\dagger} \otimes I)\text{multi-}U \left(\sum_{k=0}^{q-1} \sqrt{\alpha_{jk}}|k\rangle \right) |j\rangle|\psi\rangle && (26)\\
=& \frac{1}{\sqrt{s_j}}(\text{multi-}B^{\dagger} \otimes I) \left(\sum_{k=0}^{q-1} \sqrt{\alpha_{jk}}|k\rangle|j\rangle U_{jk}|\psi\rangle \right) && (27)\\
=& \frac{1}{s_j}|0\rangle|j\rangle \left(\sum_{k=0}^{q-1} \alpha_{jk} U_{jk}|\psi\rangle \right) + \sqrt{\gamma_j}|\Phi_j^{\perp}\rangle && (28)\\
=& \frac{1}{s_j}|0\rangle|j\rangle A_j|\psi\rangle + \sqrt{\gamma_j}|\Phi_j^{\perp}\rangle, && (29)
\end{aligned}$$

where $|\Phi_j^{\perp}\rangle$ is a state satisfying $(|0\rangle\langle 0| \otimes I \otimes I)|\Phi_j^{\perp}\rangle = 0$ and γ_j is some normalization factor.

Up to this point, if the indicator register were measured and $|0\rangle$ were observed as the "success" case as in the standard LCU method, then the state of the purifier and the system register collapses to $|j\rangle A_j|\psi\rangle$. However, this is not a meaningful quantum sate, as it only captures one Kraus operator of a quantum map. Now we use this specially designed quantum state $|\mu\rangle$ to obtain the desired purification state. We use the superposition $|\mu\rangle$ instead of $|j\rangle$ in the second register then, by linearity, we have

$$W|0\rangle|\mu\rangle|\psi\rangle = \sqrt{p}|0\rangle \left(\sum_{j=0}^{m-1} |j\rangle A_j|\psi\rangle \right) + \sqrt{1-p}|\Phi^{\perp}\rangle, \quad (30)$$

where $(|0\rangle\langle 0| \otimes I \otimes I)|\Phi^{\perp}\rangle = 0$ and $p = \frac{1}{\sum_{j=0}^{m-1} s_j^2}$. ◀

4 Overview of the main result, Theorem 1

In this section we briefly sketch how to apply our new LCU method in order to prove our main result, Theorem 1. The overall structure is similar to that in [2] and [3], with the main novel ingredient being our variant of the LCU construction (explained in section 3) and also a variant of oblivious amplitude amplification for isometries. For clarity, the details are organized into section 4 of the full version of this paper [10], whose content is summarized as:

1. In Sec. 4.1 of [10], we describe a simple mapping $\mathcal{M}_\delta$ in terms of Kraus operators that are based on the operators in $\mathcal{L}$. For small δ, $\mathcal{M}_\delta$ is a good approximation of $e^{\mathcal{L}\delta}$.
2. In Sec. 4.2 of [10], we show how to simulate the mapping $\mathcal{M}_\delta$ in the sense of Lemma 3, with success probability parameter $1 - O(\delta)$.
3. In Sec. 4.3 of [10], we show how to combine r simulations of $\mathcal{M}_{O(1/r)}$ so as to obtain cumulative success probability parameter $1/4$. Conditional on success, this produces a good approximation of constant-time Lindblad evolution.
4. In Sec. 4.4 of [10], we show how to apply a modified version of oblivious amplitude amplification to unconditionally simulate an approximation of constant-time Lindblad evolution.
5. In Sec. 4.5 of [10], we show how to reduce the number of multiplexed Pauli gates by a concentration bound on the amplitudes associated with nontrivial Pauli gates.
6. In Sec. 4.6 of [10], we bound the total number of gates and combine the simulations for segments in order to complete the proof of Theorem 1.

Acknowledgments. We thank Andrew Childs, Patrick Hayden, Martin Kliesch, Tongyang Li, Hans Massen, Barry Sanders, and Rolando Somma for helpful discussions.

References

1 D. Aharonov and A. Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 20–29, 2003. doi:10.1145/780542.780546.

2 D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma. Exponential improvement in precision for simulating sparse Hamiltonians. In *Proceedings of the 46th ACM Symposium on Theory of Computing*, pages 283–292, 2014. arXiv:arXiv:1312.1414, doi:10.1145/2591796.2591854.

3 D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Phys. Rev. Lett.*, 114:090502, Mar 2015. doi:10.1103/PhysRevLett.114.090502.

4 D. W. Berry, A. M. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Proceedings of FOCS 2015*, 2015. arXiv:arXiv:1501.01715.

5 D. W. Berry, R. Cleve, and S. Gharibian. Gate-efficient discrete simulations of continuous-time quantum query algorithms. *Quantum Information and Computation*, 14(1–2):1–30, 2014. arXiv:arXiv:1211.4637.

6 D. W. Berry and L. Novo. Corrected quantum walk for optimal Hhamiltonian simulation, 2016. arXiv:1606.03443.

7 R. Di Candia, J. S. Pedernales, A. del Campo, E. Solano, and J. Casanova. Quantum simulation of dissipative processes without reservoir engineering. *Sci. Rep.*, 5:9981, 2015.

8 A. M. Childs. *Quantum information processing in continuous time.* PhD thesis, Massachusetts Institute of Technology, 2014.

9 A. M. Childs and T. Li. Efficient simulation of sparse markovian quantum dynamics, 2016. arXiv:1611.05543.

10 Richard Cleve and Chunhao Wang. Efficient quantum algorithms for simulating lindblad evolution. *arXiv preprint arXiv:1612.09512*, 2016.

11 R. Dorner, J. Goold, and V. Vedral. Towards quantum simulations of biological information flow. *Interface focus*, page rsfs20110109, 2012.

12 R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6–7):467–488, 1982.

13 V. Gorini, A. Kossakowski, and E. C. G. Sudarshan. Completely positive dynamical semigroups of n-level systems. *Journal of Mathematical Physics*, 17:821–825, 1976.

14 S. F. Huelga and M. B. Plenio. Vibrations, quanta and biology. *Contemporary Physics*, 54(4):181–207, 2013.

15 M. J. Kastoryano and F. G. S. L. Brandao. Quantum gibbs samplers: the commuting case. *Communications in Mathematical Physics*, 344(3):915–957, 2016.

16 M. J. Kastoryano, F. Reiter, and A. S. Sørensen. Dissipative preparation of entanglement in optical cavities. *Physical review letters*, 106(9):090502, 2011.

17 M. Kliesch, T. Barthel, C. Gogolin, M. Kastoryano, and J. Eisert. Dissipative quantum church-turing theorem. *Physical review letters*, 107(12):120501, 2011.

18 R. Kothari. *Efficient algorithms in quantum query complexity*. PhD thesis, University of Waterloo, 2014.

19 B. Kraus, H. P. Büchler, S. Diehl, A. Kantian, A. Micheli, and P. Zoller. Preparation of entangled states by quantum markov processes. *Physical Review A*, 78(4):042307, 2008.

20 A. J. Leggett, S. Chakravarty, A. T. Dorsey, M. P. A. Fisher, A. Garg, and W. Zwerger. Dynamics of the dissipative two-state system. *Reviews of Modern Physics*, 59(1):1, 1987.

21 G. Lindblad. On the generators of quantum dynamical systems. *Communications in Mathematical Physics*, 48:119–130, 1976.

22 S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

23 G. H. Low and I. L. Chuang. Optimal Hamiltonian simulation by quantum signal processing, 2016. arXiv:1606.02685.

24 E. Magesan, D. Puzzuoli, C. E. Granade, and D. G. Cory. Modeling quantum noise for efficient testing of fault-tolerant circuits. *Physical Review A*, 87(1):012324, 2013.

25 V. May and O. Kühn. *Charge and energy transfer dynamics in molecular systems*. John Wiley & Sons, 2008.

26 S. Mostame, P. Rebentrost, A. Eisfeld, A. J. Kerman, D. I. Tsomokos, and A. Aspuru-Guzik. Quantum simulator of an open quantum system using superconducting qubits: exciton transport in photosynthetic complexes. *New Journal of Physics*, 14(10):105013, 2012.

27 A. Nitzan. *Chemical dynamics in condensed phases: relaxation, transfer and reactions in condensed molecular systems*. Oxford university press, 2006.

28 A. Patel and A. Priyadarsini. Optimization of quantum hamiltonian evolution: from two projection operators to local hamiltonians. *International Journal of Quantum Information*, page 1650027, 2016.

29 F. Reiter, D. Reeb, and A. S. Sørensen. Scalable dissipative preparation of many-body entanglement. *Physical Review Letters*, 117(4):040501, 2016.

30 M. Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, 1991.

31 F. Verstraete, M. M. Wolf, and J. I. Cirac. Quantum computation and quantum-state engineering driven by dissipation. *Nature physics*, 5(9):633–636, 2009.

32 U. Weiss. *Quantum dissipative systems*. World scientific, 2012.

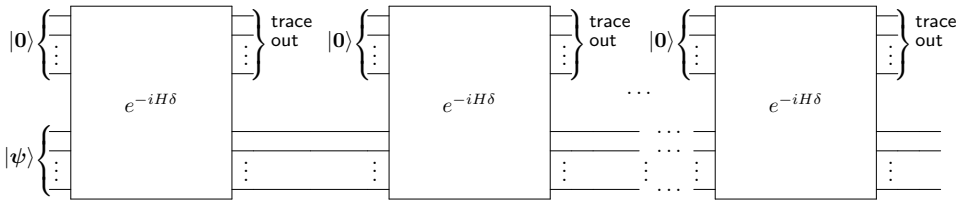


Figure 3 N-stage ϵ-precision discretization of the trajectory resulting from $\mathcal{L}$. For each $k \in \{1, \ldots, N\}$, after k stages, the channel should be within ϵ of $\exp\left(\frac{kT}{N}\mathcal{L}\right)$.

A Cost of expressing Lindblad evolution as Hamiltonian evolution

Let $\mathcal{L}$ be a Lindbladian acting on an n-qubit register $\mathcal{H}$ over a time interval $[0, T]$. For each initial state, $\mathcal{L}$ associates a *trajectory*, consisting of a density operator $\rho(t)$ for each $t \in [0, T]$. Here we show that if this is simulated by Hamiltonian evolution in a larger system with an ancillary register that is continually reset (expressed as a limiting case when $N \to \infty$ in the process illustrated in Figure 3) then the total evolution time for this Hamiltonian can be necessarily infinite.

▶ **Definition 4.** Define an *N-stage ϵ-precision discretization of $\mathcal{L}$ for interval* $[0, T]$ as an ancillary register $\mathcal{K}$, a Hamiltonian H (with $\|H\| = 1$) acting on the joint system $\mathcal{K} \otimes \mathcal{H}$, and $\delta \geq 0$ such that the channel $\mathcal{N}_{H\delta}$ defined as

$$\mathcal{N}_{H\delta}[\rho] = \mathrm{Tr}_{\mathcal{K}}\left(e^{-iH\delta}(|0\rangle\langle 0| \otimes \rho)e^{iH\delta}\right) \tag{31}$$

has the following property. $\mathcal{N}_{H\delta}$ approximates evolution under $\mathcal{L}$ in the sense that, for each $j \in \{1, \ldots, N\}$,

$$\left\|(\mathcal{N}_{H\delta})^k - \exp\left(\tfrac{kT}{N}\mathcal{L}\right)\right\|_\diamond \leq \epsilon. \tag{32}$$

That is, the N points generated by $\mathcal{N}_{H\delta}, (\mathcal{N}_{H\delta})^2, \ldots, (\mathcal{N}_{H\delta})^N$ approximate the corresponding points on the trajectory determined by $\mathcal{L}$.

Our lower bound is for the *amplitude damping process* on a 1-qubit system is the time-evolution described by the Lindbladian $\mathcal{L}$, where

$$\mathcal{L}[\rho] = L\rho L^\dagger - \tfrac{1}{2}(L^\dagger L\rho + \rho L^\dagger L), \tag{33}$$

and $L = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$.

▶ **Theorem 5.** *Any $\frac{1}{4}$-precision N-stage approximation of the amplitude damping process over the time interval $[0, \ln 2]$ has the property that the total evolution time of H is $\Omega(\sqrt{N})$. (Note that this lower bound is independent of the dimension of the ancillary system.)*

To prove Theorem 5, we first prove the following *Local Hamiltonian Approximation* lemma. This concerns a scenario where H is a Hamiltonian acting on a joint system of two registers, a system register $\mathcal{H}$ and an ancillary register $\mathcal{K}$, and where $\mathcal{K}$ is traced out after this evolution. Informally, the lemma states that, if the initial state is a product state and the evolution time is short, then this process can be approximated by the evolution of another Hamiltonian G that acts on $\mathcal{H}$ alone. This is illustrated in figure 4.

Figure 4 The Local Hamiltonian Approximation Lemma. The first register is d-dimensional, the second register contains n qubits, and the approximation is within $O(\delta^2)$ (independent of d and n).

▶ **Lemma 6** (Local Hamiltonian approximation). *Let $\mathcal{H}$ be an n-qubit register and $\mathcal{K}$ a d-dimensional register. Let H be a Hamiltonian (with $\|H\| = 1$) acting on the joint system $\mathcal{K} \otimes \mathcal{H}$. Define the n-qubit channel $\mathcal{N}_{H\delta}$ as*

$$\mathcal{N}_{H\delta}[\rho] = \mathrm{Tr}_{\mathcal{K}}\big(e^{-iH\delta}(|0\rangle\langle 0| \otimes \rho)e^{iH\delta}\big). \tag{34}$$

Then there exists a Hamiltonian G (with $\|G\| = 1$), acting on $\mathcal{H}$ alone, such that $\mathcal{N}_{G\delta}$ defined as

$$\mathcal{N}_{G\delta}[\rho] = e^{-iG\delta}\rho\, e^{iG\delta} \tag{35}$$

satisfies $\|\mathcal{N}_{H\delta} - \mathcal{N}_{G\delta}\|_1 \in O(\delta^2)$. (The notation $\|\cdot\|_1$ indicates the trace-induced norm, which is sufficient for our purposes because our application is a lower bound.)

Proof. Viewing H as a $d \times d$ block matrix, we have

$$H = \sum_{j=0}^{d-1}\sum_{k=0}^{d-1} |j\rangle\langle k| \otimes H_{jk} \tag{36}$$

and we refer to H_{jk} as the (j,k) *block*. Define D as the diagonal blocks of H, namely

$$D = \sum_{j=0}^{d-1} |j\rangle\langle j| \otimes H_{jj}, \tag{37}$$

and set $J = H - D$ (the off-diagonal blocks). Note that $\|D\|, \|J\| \leq 1$ and $\|e^{-iH\delta} - e^{-iD\delta}e^{-iJ\delta}\| \leq \delta^2$, for $\delta > 0$, which permits us to consider the effect of J and D separately.

Now consider the state $e^{-iJ\delta}|0\rangle\otimes|\psi\rangle$. We will show that, if the measurement corresponding to projectors $|0\rangle\langle 0|$ and $I - |0\rangle\langle 0|$ is performed on register $\mathcal{K}$, then the residual state has trace distance $O(\delta^2)$ from $|0\rangle \otimes |\psi\rangle$. Since the $(0,0)$ block of J is 0,

$$J\delta\,|0\rangle \otimes |\psi\rangle = \delta'|\Psi^{\perp}\rangle, \tag{38}$$

where $|\Psi^{\perp}\rangle$ is a state such that $(|0\rangle\langle 0| \otimes I)|\Psi^{\perp}\rangle = 0$ and $0 \leq \delta' \leq \delta$. Therefore,

$$e^{-iJ\delta}|0\rangle \otimes |\psi\rangle = \sum_{r=0}^{\infty} \frac{(-iJ\delta)^r}{r!}|0\rangle \otimes |\psi\rangle \tag{39}$$

$$= |0\rangle \otimes |\psi\rangle - i\delta'|\Psi^{\perp}\rangle + \delta''|\Phi\rangle, \tag{40}$$

where $0 \leq \delta'' \leq e^{\delta} - 1 - \delta \in O(\delta^2)$. It follows that, if the above measurement is performed on register $\mathcal{K}$, then the probability of measurement outcome $I - |0\rangle\langle 0|$ is at most $(\delta')^2 + (\delta'')^2 \in O(\delta^2)$. This implies that the state when register $\mathcal{K}$ of $e^{-iJ\delta}|0\rangle \otimes |\psi\rangle$ is traced out, namely

$$\mathrm{Tr}_{\mathcal{K}}\big(e^{-iJ\delta}(|0\rangle\langle 0| \otimes |\psi\rangle\langle\psi|)e^{iJ\delta}\big), \tag{41}$$

has trace distance $O(\delta^2)$ from the original state $|\psi\rangle\langle\psi|$.

Therefore, for states of the form $|0\rangle \otimes |\psi\rangle$, the operation $e^{-iH\delta}$ can be approximated by $e^{-iD\delta}$ at the cost of an error of $O(\delta^2)$ in trace distance. The result follows by setting $G = H_{00}$ (the $(0,0)$ block of D). ◀

Proof of Theorem 5. It is straightforward to check that, starting with the initial state $|1\rangle\langle 1|$ and evolving by the amplitude damping process for time $T = \ln 2$ produces the maximally mixed state.

Consider any $\frac{1}{4}$-precision N-stage discretization of this process, with Hamiltonian H and $\delta > 0$. We can apply the Local Hamiltonian Approximation Lemma (Lemma 6) to approximate each of the N evolutions of H with evolution by a Hamiltonian G that is local to the qubit. The result is unitary evolution of the qubit that approximates the amplitude damping process within trace distance error at most $O(N\delta^2)$.

Unitary evolution applied to $|1\rangle\langle 1|$ results in a pure state, and the trace distance between any pure state and the maximally mixed state is $\frac{1}{2}$. Therefore, to avoid a contradiction, we must have $N\delta^2 \in \Omega(1)$, which implies that $\delta \in \Omega(1/\sqrt{N})$. Therefore, the total evolution time of H is $N\delta \in \Omega(\sqrt{N})$. ◀

Controlled Quantum Amplification*

Cătălin Dohotaru[1] and Peter Høyer[2]

1 **Department of Computer Science, University of Calgary, Calgary, Canada**
cdohotaru@gmail.com
2 **Department of Computer Science, University of Calgary, Calgary, Canada**
hoyer@ucalgary.ca

Abstract

We propose a new framework for turning quantum search algorithms that decide into quantum algorithms for finding a solution. Consider we are given an abstract quantum search algorithm A that can determine whether a target g exists or not. We give a general construction of another operator U that both determines and finds the target, whenever one exists. Our amplification method at most doubles the cost over using A, has little overhead, and works by controlling the evolution of A. This is the first known general framework to the open question of turning abstract quantum search algorithms into quantum algorithms for finding a solution.

We next apply the framework to random walks. We develop a new classical algorithm and a new quantum algorithm for finding a unique marked element. Our new random walk finds a unique marked element using H update operations and $1/\epsilon$ checking operations. Here H is the hitting time, and ϵ is the probability that the stationary distribution of the walk is in the marked state. Our classical walk is derived via quantum arguments. Our new quantum algorithm finds a unique marked element using $\sqrt{H}$ update operations and $\sqrt{1/\epsilon}$ checking operations, up to logarithmic factors. This is the first known quantum algorithm being simultaneously quadratically faster in both parameters. We also show that the framework can simulate Grover's quantum search algorithm, amplitude amplification, Szegedy's quantum walks, and quantum interpolated walks.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Quantum algorithms, quantum walks, random walks, quantum search

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.18

1 Introduction

Grover's search algorithm [15], amplitude amplification [8], and quantum walks [3, 30] are highly successful methodologies for searching in quantum algorithms. They are used in search problems in which we are given some unitary operator W as a black-box. We start in some initial state $|\text{init}\rangle$ which is a $(+1)$-eigenvector of the unitary W. Our goal is to produce some unknown target state $|g\rangle$. We are given a reflection[1] operator $\mathsf{G} = 1 - 2|g\rangle\langle g|$ that permits us to distinguish the target state $|g\rangle$ from any other orthogonal state. Our task is to construct an algorithm that evolves the initial state $|\text{init}\rangle$ into a state that has constant overlap with

* This work has been supported in part by the Canadian Institute for Advanced Research (CIFAR) and Canada's Natural Sciences and Engineering Research Council (NSERC).

[1] To simplify later calculations, we define the operator G as the reflection about the subspace orthogonal to $|g\rangle$.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 18; pp. 18:1–18:13

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

the target state $|g\rangle$, using the operator W. The standard approach in quantum algorithmics for doing so is to use operator $\mathsf{A} = \mathsf{W} \cdot \mathsf{G}$ [5].

Much work on search problems on quantum computers has been specifically developed within quantum walks. The operator W is often derived from a random walk, in which case it is referred to as the *walk* or *update* operator. The cost of the quantum search algorithm is then typically phrased in terms of the spectral gap δ or the hitting time of the associated random walk.

The study of quantum walks has been a highly successful and active line of research, with recent results such as a quantum algorithm for triangle finding [13] using only $O(n^{5/4})$ queries. Other applications of quantum walks include verification of matrix products [9], testing group commutativity [27], formula evaluation [4], subgraph finding [10], triangle finding [26, 13, 14], and 3-distinctness [7].

These and other applications were found after the seminal works of Ambainis [3] and Szegedy [30]. Ambainis [3] gave a quantum walk for element distinctness, and Szegedy [30] gave a general method for obtaining an quantum search algorithm from any symmetric random walk [30]. Magniez et al. [25] gave a quantum algorithm that finds a unique marked element for any reversible random walk[2] of cost in the order of $\mathsf{S} + \sqrt{1/(\epsilon\delta)}\mathsf{U} + \sqrt{1/\epsilon}\mathsf{C}$. Here S, U, and C are the setup, update and checking costs of the quantum walk, δ the spectral gap of the walk, and ϵ the probability that the stationary state is in the marked state [29, 28].

Magniez et al. [23] gave a quantum algorithm that finds a unique marked element for any state-transitive random walk of cost in the order of $\mathsf{S} + \sqrt{H}\mathsf{U} + \sqrt{H}\mathsf{C}$. Here H is the hitting time of the random walk. Krovi et al. [22] introduced the novel idea of interpolating walks. Krovi et al. show in [20, 21] that interpolated walks can find a marked element for any reversible random walk. Their algorithm works for multiple marked elements and has cost in the order of $\mathsf{S} + \sqrt{H^+}\mathsf{U} + \sqrt{H^+}\mathsf{C}$, where H^+ is a quantity introduced in [21] and referred to as the extended hitting time. When there is a unique marked element, the extended hitting time and hitting time coincide, $H^+ = H$. When there are multiple marked elements, the extended hitting time is bounded by $H \leq H^+ \leq \frac{1}{\epsilon\delta}$. Excellent surveys on quantum walks, their history and applications, include [2, 18, 29, 32, 28].

In this work, we propose a new framework for the general setting of search problems. Our framework does *not* require that the operator W is derived from a random walk and it makes no explicit use of properties of quantum walks or random walks. The most obvious application of our framework is naturally to random walks, but is not limited to such cases. We will assume that W has only real entries, and that our target $|g\rangle$ has real coordinates in a canonical orthogonal basis for the space acted upon by W. This assumption is fulfilled in all the applications we consider here, including quantum walks.

Our framework consists of several parts. We give a new generic quantum algorithm for solving search problems. Our algorithm is based on a circuit U which controls the search, and we refer to this process as *controlled quantum amplification*. We prove that whenever the operator $\mathsf{A} = \mathsf{W} \cdot \mathsf{G}$ determines whether a target $|g\rangle$ exists or not, in some number of iterations T, with constant success probability, then our circuit U both determines and finds the target using at most $2T$ iterations, with constant success probability.

We apply our analysis to quantum walks and prove that we can obtain a quadratic speedup for reversible random walks. We show that our framework can simulate quantum interpolated

[2] A Markov chain P with a unique stationary distribution $\pi = (\pi_x)$ is said to be *reversible* if $\pi_x P_{y,x} = \pi_y P_{x,y}$ for all states x, y. All Markov chains obtained from a random walk on undirected graphs are reversible. We will below use the wording "random walk" and "Markov chain" interchangeably.

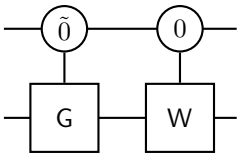


Figure 1 Our circuit U for controlled quantum amplification, expressed in terms of the reflection operator G and the unitary W, here given in its simplest form. We apply circuit U successively to the initial state $|0\rangle|\overline{\text{init}}\rangle$, say T times, thus producing the final state $\mathsf{U}^T|0\rangle|\overline{\text{init}}\rangle$, which we measure. If the outcome of the measurement is $|\tilde{1}\rangle|g\rangle$, the circuit has successfully found the marked state g.

walks [21], thus eliminating the need for running an interpolation of a random walk and its absorbing analog. We prove a relation between the operator W and the operator A, and we use this relation to construct a new quantum algorithm that, up to a logarithmic factor, finds a unique marked element in cost $\mathsf{S}+\sqrt{H}\mathsf{U}+\sqrt{1/\epsilon}\mathsf{C}$. This is superior to the existing algorithms. We also use this relation to construct a new *classical* algorithm that finds a unique marked element in cost $\mathsf{S}+H\mathsf{U}+1/\epsilon\mathsf{C}$. Our classical walk is derived using quantum arguments.

2 Controlled quantum amplification

In a quantum search problem, we are given an arbitrary unitary operator W, through which we would like to extract some unknown target state $|g\rangle$. The operator W can be given as a black box, which we can apply to any state $|\psi\rangle$, producing the state $\mathsf{W}|\psi\rangle$. An application of W has some cost, which is called the *update cost*.

We are also given a reflection operator $\mathsf{G}=1-2\Pi_G$, where $\Pi_G=|g\rangle\langle g|$ is a projection on the target state $|g\rangle$. Operator G permits us to distinguish the target state $|g\rangle$ from any other orthogonal state. The operator G can be given as a black box as well. An application of G has some cost, which is called the *checking cost*.

Our first step is to prepare the initial state $|\text{init}\rangle$. Preparing this state has a cost, which is called the *setup cost*. We then measure whether the initial state contains the target state or not by applying the measurement $\{\Pi_G, 1-\Pi_G\}$. The probability by which we measure $|g\rangle$ is some $\epsilon=\sin^2(\theta)$ where $\sin(\theta)=\langle g|\text{init}\rangle$. Let $|g\rangle$ and this initial angle θ be so that $0\le\theta\le\pi/2$. If $\theta=0$, our initial success probability is zero and the quantum search algorithm $\mathsf{A}=\mathsf{W}\cdot\mathsf{G}$ will not amplify this. If $\theta=\pi/2$, our initial state *is* the target state and there is nothing to be amplified. We shall therefore assume that $0<\theta<\pi/2$. Typically θ is very close to zero, corresponding to that the non-amplified success probability ϵ is small.

If our initial measurement yields the outcome $|g\rangle$, we terminate the algorithm as we have produced our target state $|g\rangle$. Otherwise, our initial state becomes $|\overline{\text{init}}\rangle=\frac{1}{\cos(\theta)}(|\text{init}\rangle-\sin(\theta)|g\rangle)$, which is orthogonal to $|g\rangle$ by construction. Thus starting with $|\overline{\text{init}}\rangle$, we then want to produce a state with large overlap with $|g\rangle$. To achieve this, we propose the following circuit U.

The circuit U acts on two registers. The first register contains a qubit which we use to control the evolution in the second register. The circuit is comprised of two operators, both of which are controlled on the state in the first register. We fix an angle $0<\tilde{\theta}<\pi/2$ and use the rotated orthogonal basis $|\tilde{0}\rangle=\cos(\tilde{\theta})|0\rangle+\sin(\tilde{\theta})|1\rangle$ and $|\tilde{1}\rangle=-\sin(\tilde{\theta})|0\rangle+\cos(\tilde{\theta})|1\rangle$. The first operator $|\tilde{0}\rangle\langle\tilde{0}|\otimes\mathsf{G}+|\tilde{1}\rangle\langle\tilde{1}|\otimes 1$ in our circuit reflects about the target state $|g\rangle$ conditional on that the control qubit is in state $|\tilde{0}\rangle$. The second operator $|0\rangle\langle 0|\otimes\mathsf{W}+|1\rangle\langle 1|\otimes 1$ applies the update W conditional on that the control qubit is in state $|0\rangle$.

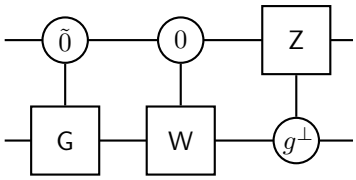


Figure 2 W with reflection.

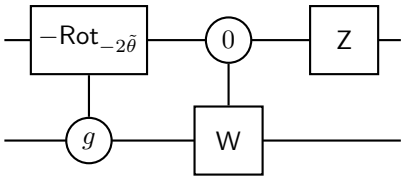


Figure 3 W with rotation.

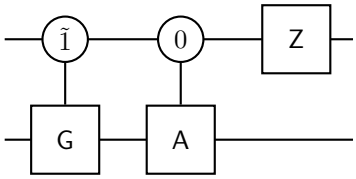


Figure 4 $\mathsf{A} = \mathsf{W} \cdot \mathsf{G}$ with reflection.

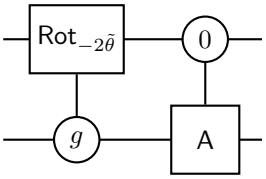


Figure 5 A with rotation.

The circuit U is parameterized by the angle $\tilde{\theta}$. The choice of $\tilde{\theta}$ has some algorithmic consequences, which we may handle by exponential searching similar to past work [8, 23]. The framework applies with no increase in asymptotic cost if we are given a multiplicatively approximate value for the initial success amplitude $\sin(\theta)$. We apply circuit U successively to the initial state $|0\rangle|\overline{\text{init}}\rangle$, say T times, thus producing the final state $\mathsf{U}^T|0\rangle|\overline{\text{init}}\rangle$, which we measure. We prove here that for an appropriately chosen value of T, the final measurement yields the target $|g\rangle$ with probability at least a constant, and we thus refer to this process as *controlled quantum amplification*.

The circuit has multiple interpretations and forms which both provide us with flexibility in terms of implementations as well as a foundation for proving properties on quantum amplification processes. We give here four circuits, all of which act equivalently to U. In Figure 2, the third gate $\mathsf{Z} \otimes (1 - |g\rangle\langle g|) + 1 \otimes |g\rangle\langle g|)$ applies the phase gate $\mathsf{Z} = |0\rangle\langle 0| - |1\rangle\langle 1|$ on the control qubit conditional on that the search space does not contain the target state $|g\rangle$. In Figures 3 and 5, the first gate rotates the ancilla qubit by an angle of $\pi - 2\tilde{\theta}$ and $-2\tilde{\theta}$, respectively, conditional on the search space contains the target state $|g\rangle$.

One of our aims is to compare our new controlled amplification circuit U with the operator $\mathsf{A} = \mathsf{W} \cdot \mathsf{G}$. The operator A has been used extensively in quantum walks and, when applied to random walks, it corresponds to an absorbing random walk [30, 3, 5]. The main limitation of the operator A is that it does not necessarily produce the target state, even when one exists. Significant research has been put into understanding when the operator A does and does not produce the target state. Our proposed controlled circuit circumvents this barrier in all generality. We prove that whenever the operator A determines whether a unique target exists or not, in some number of iterations, then our circuit U both determines and finds the target in the same asymptotic number of iterations.

Controlled quantum computations have been successfully applied in quantum computing dating back to at least the notion of phase kick-back [11]. Tulsi used a quantum walk with controlled operators for the problem of finding a unique target $|g\rangle$ on a grid [31]. His algorithm finds a unique target in cost $O(\sqrt{n \log n})$, which is quadratically smaller than the classical hitting time of $\Theta(n \log n)$ [1]. The grid graph is a two-dimensional torus of size $\sqrt{n} \times \sqrt{n}$ and has been a notoriously hard case for quantum searching because all of its edges are local. Magniez, Nayak, Richter, and Santha [23] extended this and gave a controlled operator that finds a unique target $|g\rangle$ for any state-transitive graph.

Our controlled amplifier does not rely on any graph-theoretic properties. We prove the general statement that whenever A determines whether a unique target state exists or not, our controlled amplifier finds the target state in asymptotically the same cost.

3 Quantum hitting times

The hitting time is a notion used in the analysis of stochastic processes such as random walks. It is the expected number of steps some stochastic process U uses to reach the target state g, starting from some appropriately defined initial distribution π. The choice of an appropriate corresponding definition of quantum hitting time is non-trivial.

Let U be any real unitary, and let $|w\rangle$ be any normalized target state with real coordinates in a canonical orthogonal basis for the space acted upon by U. The possible eigenvalues for U are $+1$, -1, and conjugated pairs $(e^{i\alpha}, e^{-i\alpha})$ for some eigenphase $0 < \alpha < \pi$. Each such pair of eigenvalues corresponds to a distinct two-dimensional subspace acted upon by U by a rotation of angle α. We order these non-trivial eigenphases of U as $0 < \alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_m < \pi$, for some $m \geq 0$. Let $|U_j^+\rangle$ and $|U_j^-\rangle$ be the conjugated eigenvectors of U corresponding to the eigenvalues $e^{i\alpha_j}$ and $e^{-i\alpha_j}$, for each $1 \leq j \leq m$. We decompose $|w\rangle$ into this ordered eigenbasis of U, as $w_0|U_0\rangle + \sum_{j=1}^{m} \left(w_j^+|U_j^+\rangle + w_j^-|U_j^-\rangle\right) + w_{-1}|U_{-1}\rangle$. Here we group all $(+1)$-eigenvectors of U into $|U_0\rangle$, and all the (-1)-eigenvectors into $|U_{-1}\rangle$. Since U and $|w\rangle$ have real components, we can choose the scalars of the eigenvectors of U such that $w_j^+ = w_j^- = w_j \in \mathbb{R}$. Given this basis, we define the quantum hitting time as follows.

▶ **Definition 1.** The *quantum hitting time* of U on $|w\rangle = w_0|U_0\rangle + \sum_{j=1}^{m} w_j\left(|U_j^+\rangle + |U_j^-\rangle\right) + w_{-1}|U_{-1}\rangle$ is

$$\mathsf{QHT}_\alpha(\mathsf{U}, |w\rangle) = \sqrt{2\sum_{j=1}^{m} |w_j|^2 \frac{1}{\alpha_j^2}}. \tag{1}$$

Since our proofs use the specifics of this definition, we mention that our definition differs from the more commonly used quantity $\mathsf{QHT}_1(\mathsf{U}, |w\rangle) = 2\sum_{j=1}^{m} |w_j|^2 \frac{1}{\alpha_j} + |w_{-1}|$. Our notion of quantum hitting time QHT_α is quadratically smaller than the classical hitting time for reversible random walks, by Szegedy's correspondence [30]. Thus, if a quantum algorithm has cost in the order of QHT_α, then it has cost quadratically smaller than the classical hitting time. For technical reasons, we need to introduce a second notion of quantum hitting time, which we refer to as the *cotangent quantum hitting time* and define as $\mathsf{QHT}_{\cot}(\mathsf{U}, |w\rangle) = \sqrt{2\sum_{j=1}^{m} |w_j|^2 \cot^2(\frac{\alpha_j}{2})}$. Our two notions of quantum hitting times QHT_α and $\mathsf{QHT}_{\cot}$ are asymptotically of the same order, as shown in Lemma 9 in the appendix. We use QHT as a shorthand for QHT_α.

4 Finding in the quantum hitting time

Our goal is to prove that the circuit U finds a target state in the quantum hitting time, stated as Corollary 6 below. We first identify the principal eigenvector of the circuit U.

▶ **Lemma 2.** *The unnormalized state* $|v_0\rangle = \sin(\tilde{\theta})\,|0, \overline{\text{init}}\rangle - \frac{\sin(\theta)}{\cos(\theta)}\,|\tilde{1}, g\rangle$ *is a* $(+1)$*-eigenvector of circuit* U.

Proof. The proof follows be considering the action of the circuit U given in Figure 1.

The first operator in the circuit reflects the state $|\tilde{0}, g\rangle$. The first term in the state $|v_0\rangle$ is orthogonal to this reflection state since $|\overline{\text{init}}\rangle$ is orthogonal to $|g\rangle$ by definition. The second term in $|v_0\rangle$ is also orthogonal to the reflection state since $|\tilde{0}\rangle$ and $|\tilde{1}\rangle$ constitute an orthonormal basis. The reflection operator thus acts trivially on $|v_0\rangle$.

The second operator in the circuit applies the operator W on the search register conditional on that the control qubit is in state $|0\rangle$. Rewrite $|v_0\rangle$ on the form

$$\frac{\sin(\tilde{\theta})}{\cos(\theta)}|0, \text{init}\rangle - \frac{\sin(\theta)}{\cos(\theta)}\cos(\tilde{\theta})\,|1, g\rangle.$$

Then each of the two terms is again invariant, and we conclude that the second operator similarly acts trivially on $|v_0\rangle$. ◀

We want this principal eigenvector to be an equally weighted superposition of our initial state $|0, \overline{\text{init}}\rangle$ and our target state $|\tilde{1}, g\rangle$. We therefore choose and fix the angle $\tilde{\theta}$ such that $0 < \tilde{\theta} < \pi/2$ and $\sin(\tilde{\theta}) = \frac{\sin(\theta)}{\cos(\theta)}$, and write the principal eigenvector on the normalized form $|U_0\rangle = \frac{1}{\sqrt{2}}\big(|0, \overline{\text{init}}\rangle - |\tilde{1}, g\rangle\big)$.

When considering quantum search problems, it is commonly assumed that the walk operator W has a unique $(+1)$-eigenvector (up to scalars), which we adapt here for convenience. The purpose of a quantum search problem is to produce a state that has large overlap with the target state $|g\rangle$. Our proposed algorithm for doing so, is to apply our circuit U successively a number of T times on the initial state $|0, \overline{\text{init}}\rangle$, producing the final state $\mathsf{U}^T|0, \overline{\text{init}}\rangle$. We show that it suffices to pick the number of iterations T to be in the order of the quantum hitting time $\mathsf{QHT}(\mathsf{U}, |\tilde{1}, g\rangle)$ of U. We divide the proof up into two cases. When W is a reflection, our circuit emulates amplitude amplification. For general operators W, we use that the principal $(+1)$-eigenvector of U has large overlap with both the initial state and the target state $|\tilde{1}, g\rangle$.

Consider first that $\mathsf{W} = 2|\text{init}\rangle\langle\text{init}| - 1$ is a reflection about the initial state. In amplitude amplification [8], we apply the operator $\mathsf{A} = \mathsf{W} \cdot \mathsf{G}$ a number of T times, thus constructing the state $|\text{final}\rangle = (\mathsf{W} \cdot \mathsf{G})^T|\text{init}\rangle$. By picking $T = \lceil \frac{\pi}{4\theta} \rceil \in \Theta(\frac{1}{\sqrt{\epsilon}})$, a measurement of the final state $|\text{final}\rangle$ yields the target state $|g\rangle$ with probability at least $1 - \epsilon$, where $\epsilon = \sin^2(\theta) = \langle g|\text{init}\rangle^2$ is the initial success probability. Amplitude amplification thus amplifies quadratically faster than classical repetition.

Now consider circuit U given in Figure 1 when $\mathsf{W} = 2|\text{init}\rangle\langle\text{init}| - 1$ is a reflection about the initial state. Then U is the product of two reflections and effectively implements a rotation in the two-dimensional subspace spanned by $|\tilde{0}, g\rangle$ and $|0, \overline{g}\rangle$, where $|\overline{g}\rangle = \frac{1}{\cos(\theta)}(|g\rangle - \sin(\theta)|\text{init}\rangle)$ is defined analogously to $|\overline{\text{init}}\rangle$. The rotational angle is 2φ, where φ is given by the inner product $\cos(\varphi) = \langle 0, \overline{g}|\tilde{0}, g\rangle = \cos(\tilde{\theta})\cos(\theta) = \sqrt{\cos(2\theta)}$. This gives us that $2\varphi \approx 2\sqrt{2}\theta$. The initial state is $|0, \overline{\text{init}}\rangle = \frac{1}{\sqrt{2}}(|U_0\rangle + |U_{\text{rot}}\rangle)$, where $|U_{\text{rot}}\rangle$ belongs to the two-dimensional rotational subspace. We pick $T = \lceil \frac{\pi}{2\sqrt{2}\theta} \rceil$, and produce the final state $\mathsf{U}^T|0, \overline{\text{init}}\rangle$, a measurement of which yields the target $|\tilde{1}, g\rangle = \frac{1}{\sqrt{2}}(|U_0\rangle - |U_{\text{rot}}\rangle)$ with probability at least $1 - O(\epsilon)$.

4.1 Finding in the quantum hitting time for general A

When W is a reflection about the initial state $|\text{init}\rangle$, the previous subsection implies that it suffices to choose T to be in the order of $\frac{1}{\theta}$, just as in amplitude amplification. For arbitrary operators W, it suffices to choose T to be in the order of the quantum hitting time $\mathsf{QHT}(\mathsf{U}, |0, \overline{\text{init}}\rangle)$ of U on the initial state $|0, \overline{\text{init}}\rangle$.

▶ **Theorem 3.** *There is an algorithm that applies* U *an expected number of order* $\mathsf{QHT}(\mathsf{U}, |0,\overline{\text{init}}\rangle)$ *times to the initial state* $|0,\overline{\text{init}}\rangle$*, and produces a final state with constant overlap with* $|\tilde{1},g\rangle$*.*

The proof is as follows. By Lemma 2, the initial state $|0,\overline{\text{init}}\rangle$ can be written as an equal superposition of the principal eigenvector $|U_0\rangle$ of U and the state $|U_{\text{rot}}\rangle = \frac{1}{\sqrt{2}}\big(|0,\overline{\text{init}}\rangle + |\tilde{1},g\rangle\big)$. The state $|U_0\rangle$ has constant overlap with the target state $|\tilde{1},g\rangle$. The state $|U_0\rangle$ is an eigenvector of U with eigenphase 0, whereas the state $|U_{\text{rot}}\rangle$ is a superposition of states with non-zero eigenphases. Following a standard argument via phase estimation [19, 11, 8, 24, 23], we can determine which is the case by successive (controlled) U applications. The primary observations are that the success probability in phase estimation can be expressed naturally in terms of our quantum hitting time QHT_α, and the controls on U can be dropped.

The theorem remains true if we replace the quantum hitting time with the effective quantum hitting time, at the expense of a drop in the overlap by at most a small constant. Here the effective quantum hitting time is the smallest number of applications of (controlled) U required to produce a final state with constant overlap with $|\tilde{1},g\rangle$. By Markov's inequality, the effective quantum hitting time is at most in the order of the quantum hitting time.

Theorem 3 provides us with an expression of the cost of U in terms of the quantum hitting time of U itself. We next relate the quantum hitting time of U to the quantum hitting times of quantum search operators A and W. Let $\epsilon = \sin^2(\theta)$ be the initial success probability, where angle $0 < \theta < \pi/2$ is so that $\sin(\theta) = |\langle g|\text{init}\rangle|$, and set angle $0 < \tilde{\theta} < \pi/2$ so that $\sin(\tilde{\theta}) = \frac{\sin(\theta)}{\cos(\theta)}$.

▶ **Theorem 4.**

$$\mathsf{QHT}(\mathsf{U}, |0,\overline{\text{init}}\rangle) = \mathsf{QHT}(\mathsf{U}, |\tilde{1},g\rangle) = \Theta\left(\frac{1}{\sqrt{\epsilon}}\mathsf{QHT}(\mathsf{W}, |\bar{g}\rangle)\right) = \Theta\big(\mathsf{QHT}(\mathsf{A}, |\overline{\text{init}}\rangle)\big).$$

Consider Theorem 4. The first equality follows since the principal $(+1)$-eigenvector $|U_0\rangle = \frac{1}{\sqrt{2}}(|0,\overline{\text{init}}\rangle - |\tilde{1},g\rangle)$ of the controlled amplifier U is an equal superposition of the starting state and the target state. To prove the remaining two equalities, we relate the computational quantity QHT to a structural quantity, an inner product.

▶ **Lemma 5.** *Let* V *be any real unitary, and let* $|w\rangle$ *be a real state that does not overlap the* $(+1)$*-eigenspace* V_+ *of* V*. Then the operator* $\mathsf{S} = \mathsf{V}\cdot(1-2|w\rangle\langle w|)$ *has a unique* $(+1)$*-eigenvector* $|+S\rangle$ *orthogonal to* V_+*, and it satisfies that* $\mathsf{QHT}(\mathsf{V}, |w\rangle) = \Theta\big(\frac{1}{|\langle w|+S\rangle|}\big)$*.*

Proof. We first decompose $|w\rangle$ into the eigenbasis of V as $|w\rangle = \sum_j w_j(|V_j^+\rangle + |V_j^-\rangle) + w_{-1}|V_{-1}\rangle$, where $w_j \in \mathbb{R}$ for all j. The states $|V_j^+\rangle$ and $|V_j^-\rangle$ are the two conjugated eigenstates for the j^{th} rotational subspace of V with eigenphases $\pm\varphi_j$. The state $|V_{-1}\rangle$ is the normalized projection of $|w\rangle$ onto the (-1)-eigenspace of V (when it exists). Then S has a $(+1)$-eigenvector on the form $|+w\rangle = |w\rangle + i|w^\perp\rangle = |w\rangle + i\sum_j w_j \cot\left(\frac{\varphi_j}{2}\right)\big(|V_j^-\rangle - |V_j^+\rangle\big)$, and it is the unique $(+1)$-eigenvector orthogonal to the trivial $(+1)$-eigenspace V_+. The norm $\|+w\|$ of $|+w\rangle$ is

$$\sqrt{1+\||w^\perp\rangle\|^2} = \sqrt{1+2\sum_j w_j^2\cot^2\left(\frac{\varphi_j}{2}\right)} = \Theta(\mathsf{QHT}_{\cot}(\mathsf{V}, |w\rangle)) = \Theta(\mathsf{QHT}(\mathsf{V}, |w\rangle)),$$

where the last equality follows by Lemma 9 given in the appendix. The normalized eigenvector is then $|+S\rangle = \frac{1}{\Theta(\mathsf{QHT}(\mathsf{V},|w\rangle))}\left(|w\rangle + i|w^\perp\rangle\right)$. ◀

Note that in Theorem 4, the last two vectors are orthogonal to the $(+1)$-eigenspaces of the respective operators, and hence Lemma 5 applies. Since the corresponding inner products

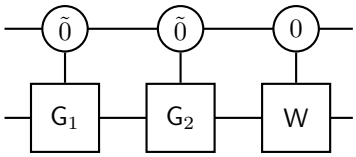


Figure 6 The circuit U rewritten to permit an analysis of its action when there are multiple targets.

are of the same order and also of the same order as the quantity $\mathsf{QHT}(\mathsf{U}, |\tilde{1}, g\rangle)$, we deduce the four quantum hitting times are of the same order, implying Theorem 4.

Theorem 4 implies that the quantum hitting time of U on $|0, \overline{\text{init}}\rangle$ is asymptotically the same as the quantum hitting time of A on $|\overline{\text{init}}\rangle$.

▶ **Corollary 6.** *There is an algorithm that applies circuit* U *an expected number in the order of* $\mathsf{QHT}(\mathsf{A}, |\overline{\text{init}}\rangle)$ *times to the initial state* $|0, \overline{\text{init}}\rangle$ *and produces a final state with constant overlap with* $|\tilde{1}, g\rangle$.

When A is a quantum walk derived from a reversible random walk P using Szegedy's construction [30], then A can detect the presence of a unique target in cost in the order of $\mathsf{QHT}(\mathsf{A}, |\overline{\text{init}}\rangle) \in O(\sqrt{\mathsf{HT}(\mathsf{P}, \{m\})})$, which is quadratically less than the hitting time of P [23, 28]. The corollary thus implies that U finds a unique target quadratically faster than classically.

5 Finding with multiple targets

We have analyzed our circuit U given in Figure 1 for a single target state $|g\rangle$. Consider now we have t targets. Let $|g_1\rangle, \ldots, |g_t\rangle$ be a set of t orthogonal states spanning this target subspace $\mathcal{G}$, and let Π_G be the projection onto $\mathcal{G}$. In our circuit U in Figure 1, consider we now have $\mathsf{G} = 1 - 2\Pi_G$. Let $|g_\pi\rangle$ be the normalized projection of $|\text{init}\rangle$ onto the target subspace. Then $\epsilon_\pi = \sin^2(\theta) = |\langle g_\pi|\text{init}\rangle|^2$ is the total probability that a measurement of the initial state would successfully produce any of the target states.

To analyze the circuit U when there are multiple targets, we again rewrite the circuit. Set $\mathsf{G}_1 = 1 - 2|g_\pi\rangle\langle g_\pi|$ and let $\mathsf{G}_2 = 1 - 2(\Pi_G - |g_\pi\rangle\langle g_\pi|)$. We can then write our circuit U on the form given in Figure 6. The rewritten circuit differs in form from our original circuit by the second gate $|\tilde{0}\rangle\langle\tilde{0}| \otimes \mathsf{G}_2 + |\tilde{1}\rangle\langle\tilde{1}| \otimes 1$. In general, this second gate changes the quantum hitting time of the circuit. For some classes of operators W, though, the second gate has no impact. In particular, for quantum walks on reversible graphs when the operator W is comprised of a reflection and a swap operator [30], we can compute an explicit expression of the complexity.

▶ **Theorem 7.** *Let* W *be a reversible quantum walk with multiple marked elements. There is an algorithm that applies* U *an expected number in the order of* $\frac{1}{\sqrt{\epsilon_\pi}} \cdot \mathsf{QHT}(\mathsf{W}, |\overline{g_\pi}\rangle)$ *times on the initial state* $|0, \overline{\text{init}}\rangle$ *and produces a final state with constant overlap with* $|\tilde{1}, g_\pi\rangle$.

Note that in this theorem, $|g_\pi\rangle$ is the normalized projection of $|\text{init}\rangle$ onto the marked subspace and ϵ_π is the total probability a measurement of $|\text{init}\rangle$ yields a marked element. The state $|\overline{g_\pi}\rangle$ is the normalized projection of $|g_\pi\rangle$ onto the subspace orthogonal to $|\text{init}\rangle$. The theorem follows by observing that the quantum hitting times on the input $|0, \overline{\text{init}}\rangle$ with or without the second gate are the same.

6 Faster algorithms for a unique marked element

Consider there is a unique marked element g. By Theorem 4, we can write $\mathsf{QHT}^2(\mathsf{A}, |\overline{\text{init}}\rangle)$ as a product of the two factors $\frac{1}{\epsilon}$ and $\mathsf{QHT}^2(\mathsf{W}, |\overline{g}\rangle)$. This decomposition permits us to device a new quantum algorithm for finding a unique marked element with constant success probability of cost in the order of

$$\mathsf{S} + \sqrt{H}\mathsf{U} + \frac{1}{\sqrt{\epsilon}}\mathsf{C}. \tag{2}$$

Here $H \in \widetilde{O}(\mathsf{QHT}^2(\mathsf{A}, |\overline{\text{init}}\rangle))$ is in the order of the square of the quantum hitting time, up to logarithmic factors. When A is the quantum walk derived from a reversible random walk P using Szegedy's construction, then $\mathsf{QHT}^2(\mathsf{A}, |\overline{\text{init}}\rangle) \in O(\mathsf{HT}(\mathsf{P}, \{g\}))$ is in the order of the hitting time of P with unique marked element g.

Since the hitting time $\mathsf{HT}(\mathsf{P}, \{g\})$ is upper bounded by $\frac{1}{\epsilon\delta}$, the cost of our algorithm is, up to logarithmic factors, upper bounded by the cost of the existing algorithms in [25, 23, 21]. Our algorithm is derived by using Theorem 4 and is based on recursive amplitude amplification [17, 16, 25]. In terms of the checking cost C, our algorithm has the same cost as amplitude amplification would have, which offers a quadratic speed-up over any classical algorithm.

We also obtain a new *classical* algorithm that has cost of order

$$\mathsf{S} + H\mathsf{U} + \frac{1}{\epsilon}\mathsf{C}. \tag{3}$$

Here $H \in O(\mathsf{HT}(\mathsf{P}, \{g\}))$ is the hitting time of the reversible walk P with unique marked element g, without logarithmic factors. This combines the best of two natural choices of random walks having cost in the order of $\mathsf{S}+H(\mathsf{U}+\mathsf{C})$ and $\mathsf{S}+\frac{1}{\epsilon}(\frac{1}{\delta}\mathsf{U}+\mathsf{C})$ (see e.g. Santha [29] for a discussion on classical search algorithms). The main structure in our classical algorithm is as follows.

1. Sample an initial vertex x according to the stationary distribution π.
2. Repeat the following of order H/E times
   a. Let x denote the current state.
   b. Check if x is marked. If so, halt and output x.
   c. Else apply the random walk P a number of E times, starting from x.
3. If the current state x is marked, output x. Otherwise output "no marked element found."

The proof is by showing that $\mathsf{QHT}^2(\mathsf{W}(\mathsf{P}^E), \{g\})$ is upper bounded by a constant when E is of order $\mathsf{QHT}^2(\mathsf{W}(\mathsf{P}), \{g\})$. This statement is effectively an example of a classical theorem derived via quantum arguments. We have not been able to find this classical algorithm discussed in the literature before, and it is to the best of our knowledge new. We neither know of a way to prove that this classical algorithm finds a marked element in cost in the order of the expression in Eq. 3 without explicitly or implicitly applying arguments resembling the arguments introduced in this paper. It is, as far as we know, the first known random walk derived through the notion of quantum walks. We refer the reader to the excellent survey by Drucker and de Wolf [12] for further examples on quantum proofs for classical theorems.

7 Simulation of quantum interpolated walks

Our controlled amplifier can be applied to arbitrary real operators W, and we prove a general bound on the cost of the amplifier given in terms of the quantum hitting time QHT_α. We

now consider operators W derived from reversible random walks. This is the broadest class of random walks for which quantum algorithms of costs in the order of the quantum hitting time have been derived. The general problem is given by a state space X on which we defined a reversible random walk P. A subset $\mathcal{M}$ of the states of X are marked (the elements of $\mathcal{M}$ correspond to the solutions to some computational problem). Our goal is to find an element of $\mathcal{M}$.

Szegedy gives a general method for constructing a quantum walk $\mathsf{W}(\mathsf{P})$ from a reversible random walk P [30, 23]. Krovi et al. give in [21] a notion of interpolation between a reversible random walk P and its corresponding absorbing walk P', which is obtained from P by replacing all the transitions from marked vertices with self-loops. The interpolation $\mathsf{P}(s)$ is between two classical walks, where we transition according to P' with some fixed probability s, and transition according to P with complementary probability $1-s$. The resulting walk $\mathsf{P}(s)$ then yields a quantum walk $\mathsf{W}(\mathsf{P}(s))$ by Szegedy's construction. We use W and $\mathsf{W}(s)$ as shorthands for $\mathsf{W}(\mathsf{P})$ and $\mathsf{W}(\mathsf{P}(s))$, respectively. Let $\mathsf{HT}(\mathsf{P}(s),\mathcal{M})$ denote the classical hitting time of $\mathsf{P}(s)$. The quantum interpolated walk introduced in [21] finds a marked element. It takes a number of steps that is in the order of $\mathsf{HT}^+(\mathsf{P},\mathcal{M})$, where $\mathsf{HT}^+(\mathsf{P},\mathcal{M})$ is defined as $\mathsf{HT}^+(\mathsf{P},\mathcal{M}) = \lim_{s\to 1}\mathsf{HT}(\mathsf{P}(s),\mathcal{M})$. This limit is well-defined and referred to as the extended hitting time [21]. We use HT and HT^+ as shorthands for $\mathsf{HT}(\mathsf{P},\mathcal{M})$ and $\mathsf{HT}^+(\mathsf{P},\mathcal{M})$ when both P and $\mathcal{M}$ are fixed.

We show that controlled quantum amplifiers can simulate quantum interpolated walks. We do so by giving a constructive embedding E_s of $\mathsf{W}(s)$ into our framework. For a given parameter s, we choose the angle $\tilde{\theta}$ so that it satisfies that $0 \le \tilde{\theta} \le \pi/2$ and that

$$\sin\tilde{\theta} = \sqrt{1-s}, \tag{4}$$

obtaining the circuit $\mathsf{U} = \mathsf{U}(\tilde{\theta})$ (see Figure 1).

Let $\mathcal{H}_{\mathcal{M}}$ be the subspace of $\mathcal{H}_{\mathsf{W}}$ with marked items in the first register. Denote by $\epsilon = \sin^2(\theta)$ the initial success probability, namely the probability that we obtain a marked state in the first register by measuring $|\mathrm{init}\rangle$ according to $\{\Pi(\mathcal{H}_{\mathcal{M}}), 1-\Pi(\mathcal{H}_{\mathcal{M}})\}$. The optimal value for $\tilde{\theta}$, which is $\tilde{\theta} = \arcsin\left(\frac{\sin(\theta)}{\cos(\theta)}\right)$, corresponds to the optimal value of s, which is given by $s = 1 - \frac{\epsilon}{1-\epsilon}$.

Denote by $\mathcal{H}_{\mathsf{W}(s)}$ the space on which the quantum walk $\mathsf{W}(s)$ acts non-trivially (which is the same as the space on which W acts non-trivially), and let $\mathcal{H}_{\mathsf{U}}$ denote the space on which $\mathsf{U}(\tilde{\theta})$ acts. We define an embedding E_s from $\mathcal{H}_{\mathsf{W}(s)}$ to a subspace of $\mathcal{H}_{\mathsf{U}}$. We define E_s on a spanning set of $\mathcal{H}_{\mathsf{W}(s)}$ and then extend it by linearity,

$$\mathsf{E}_s \begin{cases} |u, p(s)_u\rangle & \mapsto & |0\rangle|u, p_u\rangle \\ |p(s)_u, u\rangle & \mapsto & |0\rangle|p_u, u\rangle \\ |m, p(s)_m\rangle & \mapsto & -|\tilde{1}\rangle|m, p_m\rangle \\ |p(s)_m, m\rangle & \mapsto & \sin\tilde{\theta}|0\rangle|p_m, m\rangle - \cos\tilde{\theta}|1\rangle|m, p_m\rangle. \end{cases} \tag{5}$$

The state $|p_x\rangle$ is a superposition of the neighbors of x and is defined by $\langle y|p_x\rangle$ being the square-root of the probability of transition from state x to state y in the random walk P. The states $|p(s)_x\rangle$ are defined similarly the random walk $\mathsf{P}(s)$.

By direct inspection, the embedding preserves inner products and is thus well-defined. Consider the following two maps from $\mathcal{H}_{\mathsf{W}(s)}$ to $\mathcal{H}_{\mathsf{U}}$ given by $\mathsf{E}_s\mathsf{W}(s)$ and $\mathsf{U}(\tilde{\theta})\mathsf{E}_s$. The first map first applies the quantum interpolated walk and then the embedding. The second map first applies the embedding and then our controlled quantum walk $\mathsf{U}(\tilde{\theta})$. These two operators act identically on each of the states given on the left hand sides in Eq. 5, and we thus conclude that $\mathsf{E}_s\mathsf{W}(s) = \mathsf{U}(\tilde{\theta})\mathsf{E}_s$, implying Theorem 8.

▶ **Theorem 8.** *Fix any* $0 \leq s < 1$. *There is an inner-product preserving map* E_s *from* $\mathcal{H}_\mathsf{W}$ *to a subspace of* $\mathcal{H}_\mathsf{U}$ *such that* $\mathsf{E}_s\mathsf{W}(s) = \mathsf{U}(\tilde{\theta})\mathsf{E}_s$.

Theorem 8 readily implies that controlled quantum walks can simulate quantum interpolated walks. Instead of running a quantum interpolated walk on its initial state $|\overline{\text{init}}\rangle$, we run a controlled quantum walk on the initial state $|0\rangle|\overline{\text{init}}\rangle = \mathsf{E}_s|\overline{\text{init}}\rangle$. Instead of measuring whether we have produced the state $|m, p(s)_m\rangle$ in a quantum interpolated walk, we run a controlled quantum walk and measure whether we have produced the state $|\tilde{1}\rangle|m, p_m\rangle = -\mathsf{E}_s|m, p(s)_m\rangle$.

By combining Theorems 7 and 8, we obtain that the extended hitting time $\mathsf{HT}^+(\mathsf{P}, \mathcal{M})$ of a reversible walk P on a marked subset $\mathcal{M}$ is in the order of $\frac{1}{\epsilon_\pi}\mathsf{QHT}^2(\mathsf{W}(\mathsf{P}), |\overline{g_\pi}\rangle)$. Further, we also re-derive the following result already shown by Ambainis and Kokainis [6]. Since $\mathsf{QHT}(\mathsf{W}, |\overline{g_\pi}\rangle) \leq \frac{1}{\sqrt{\delta}}$, we get that the extended hitting time is never more than a factor of $1/\delta$ larger than the hitting time, and that the extended hitting time $\mathsf{HT}^+(\mathsf{P}, \mathcal{M})$ is in the order of $\frac{1}{\epsilon_\pi \delta}$ for all marked subsets $\mathcal{M}$. Here δ is the spectral gap of the random walk P.

8 Concluding remarks

An quantum search algorithm takes two ingredients: an operator W and a reflection operator G. The standard method in quantum search algorithms is to apply the composed operator $\mathsf{A} = \mathsf{W} \cdot \mathsf{G}$. This method permits one to distinguish between the case when there is a marked state and the case when there are none. It does not in general produce a marked state, even when one exists.

We have here proposed a general method for amplifying the success probability of quantum search algorithms. The method applies readily to arbitrary (real) operators W, including operators derived from random walks. Our circuit U can be based equally well on either of the two operators W or A, depending on the application in mind. We prove that the controlled amplifier U finds a unique marked element in the same asymptotic cost as the standard circuit A determines whether one exists or not. We then prove and use properties of the controlled amplifier U to simulate amplitude amplification and interpolated walks, and to derive a new quantum and classical algorithm. Up to logarithmic factors, the costs are in the order of $\mathsf{S} + \sqrt{H}\mathsf{U} + 1/\sqrt{\epsilon}\mathsf{C}$ and $\mathsf{S} + H\mathsf{U} + 1/\epsilon\mathsf{C}$, respectively. Both algorithms improve upon the best known quantum and classical algorithms.

References

1 D. Aldous and J. Fill. Reversible Markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/~aldous/RWG/book`.

2 A. Ambainis. Quantum walks and their algorithmic applications. *International Journal of Quantum Information*, 1:507–518, 2003. `arXiv:quant-ph/0403120`.

3 A. Ambainis. Quantum walk algorithm for element distinctness. In *45th IEEE Symposium on Foundations of Computer Science*, FOCS'04, pages 22–31, 2004. `doi:10.1109/FOCS.2004.54`.

4 A. Ambainis, A. M. Childs, B. Reichardt, R. Špalek, and S. Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM Journal on Computing*, 39:2513–2530, 2010. `doi:10.1109/FOCS.2007.57`.

5 A. Ambainis, J. Kempe, and A. Rivosh. Coins make quantum walks faster. In *16th ACM Symposium on Discrete Algorithms*, SODA'05, pages 1099–1108, 2005. `arXiv:quant-ph/0402107`.

6 A. Ambainis and M. Kokainis. Analysis of the extended hitting time and its properties. *Poster presented at QIP 2015*, 2015.

7 A. Belovs, A. M. Childs, S. Jeffery, R. Kothari, and F. Magniez. Time-efficient quantum walks for 3-distinctness. In *40th International Colloquium on Automata, Languages, and Programming*, ICALP'13, pages 105–122, 2013. doi:10.1007/978-3-642-39206-1_10.

8 G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002. arXiv:quant-ph/0005055.

9 H. Buhrman and R. Špalek. Quantum verification of matrix products. In *17th ACM-SIAM Symposium on Discrete Algorithms*, SODA'06, pages 880–889, 2006. arXiv:quant-ph/0409035.

10 A. M. Childs and R. Kothari. Quantum query complexity of minor-closed graph properties. In *28th Symposium on Theoretical Aspects of Computer Science*, STACS'11, pages 661–672, 2011. doi:10.4230/LIPIcs.STACS.2011.661.

11 R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London, Series A*, 454:339–354, 1998. arXiv:quant-ph/9708016.

12 A. Drucker and R. de Wolf. *Quantum Proofs for Classical Theorems*. Number 2 in Graduate Surveys. Theory of Computing Library, 2011. doi:10.4086/toc.gs.2011.002.

13 F. Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. In *55th IEEE Symposium on Foundations of Computer Science*, FOCS'14, pages 216–225, 2014. doi:10.1109/FOCS.2014.31.

14 F. Le Gall and S. Nakajima. Quantum algorithm for triangle finding in sparse graphs. In *15th Asian Quantum Information Science Conference*, AQIS'15, 2015. arXiv:1507.06878.

15 L. K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79:325–328, 1997. doi:10.1103/PhysRevLett.79.325.

16 P. Høyer and R. de Wolf. Improved quantum communication complexity bounds for disjointness and equality. In *19th Symp. on Theoretical Aspects of Computer Science*, STACS'02, pages 299–310, 2002. doi:10.1007/3-540-45841-7_24.

17 P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. In *30th International Colloquium on Automata, Languages and Programming*, ICALP'03, pages 291–299, 2003. doi:10.1007/3-540-45061-0_25.

18 J. Kempe. Quantum random walks: An introductory overview. *Contemporary Physics*, 44(4):307–327, 2003. doi:10.1080/0010715103100011077 6.

19 A. Kitaev. Quantum measurements and the abelian stabilizer problem, 1995. arXiv:quant-ph/9511026.

20 H. Krovi, F. Magniez, M. Ozols, and J. Roland. Finding is as easy as detecting for quantum walks. In *37st International Colloquium on Automata, Languages and Programming*, ICALP'10, pages 540—-551, 2010. arXiv:1002.2419v1.

21 H. Krovi, F. Magniez, M. Ozols, and J. Roland. Quantum walks can find a marked element on any graph. *Algorithmica*, 74:851–907, February 2016. doi:10.1007/s00453-015-9979-8.

22 H. Krovi, M. Ozols, and J. Roland. Adiabatic condition and the quantum hitting time of Markov chains. *Physical Review A*, 82:022333, 2010. doi:10.1103/PhysRevA.82.022333.

23 F. Magniez, A. Nayak, P. Richter, and M. Santha. On the hitting times of quantum versus random walks. *Algorithmica*, 63(1):91–116, 2012. doi:10.1007/s00453-011-9521-6.

24 F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via quantum walk. *39th ACM Symposium on Theory of Computing*, pages 575–584, 2007. doi:10.1137/090745854.

25 F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, Jan 2011. doi:10.1137/090745854.

26 F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 27:413–424, 2007. doi:10.1137/050643684.

27 A. Nayak and F. Magniez. Quantum complexity of testing group commutativity. *Algorithmica*, 48:221–232, 2007. doi:10.1007/s00453-007-0057-8.

28 A. Nayak, P. C. Richter, and M. Szegedy. Quantum analogs of markov chains. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 1–10. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. doi:10.1007/978-3-642-27848-8_302-2.

29 M. Santha. Quantum walk based search algorithms. In *International Conference on Theory and Applications of Models of Computation*, pages 31–46, 2008. doi:10.1007/978-3-540-79228-4_3.

30 M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *45th IEEE Symposium on Foundations of Computer Science*, FOCS'04, pages 32–41, 2004. doi:10.1109/FOCS.2004.53.

31 A. Tulsi. Faster quantum walk algorithm for the two dimensional spatial search. *Physical Review A*, 78:012310, 2008. doi:10.1103/PhysRevA.78.012310.

32 S. E. Venegas-Andraca. Quantum walks: A comprehensive review. *Quantum Information Processing*, 11(5):1015–1106, 2012. arXiv:1201.4780.

A A lemma

The following lemma, referenced in Section 3, shows that the two quantum hitting times, $\mathsf{QHT}_{\cot}$ and QHT_{α} are of the same asymptotic order.

▶ **Lemma 9.** *For any real unitary* U *and any real state* $|w\rangle$,

$$\mathsf{QHT}_{\cot}(\mathsf{U},|w\rangle) \le 2\,\mathsf{QHT}_{\alpha}(\mathsf{U},|w\rangle) \le \sqrt{2} + \mathsf{QHT}_{\cot}(\mathsf{U},|w\rangle).$$

Proof. Since $\sin x < x < \tan x$ for any $x \in (0,\pi/2)$,

$$\cot^2 x < \frac{1}{x^2} < 1 + \cot^2 x. \tag{6}$$

By the first inequality in Eq. 6,

$$\mathsf{QHT}_{\cot}(\mathsf{U},|w\rangle) = \sqrt{2\sum_{j=1}^{m} |w_j|^2 \cot^2\left(\frac{\alpha_j}{2}\right)} \le \sqrt{2\sum_{j=1}^{m} |w_j|^2 \left(\frac{4}{\alpha_j^2}\right)} = 2\,\mathsf{QHT}_{\alpha}(\mathsf{U},|w\rangle),$$

proving the first inequality. Since the eigenphases α_j of U belong to $(0,\pi)$, the half angles $\alpha_j/2$ belong to the interval $(0,\pi/2)$. By the second inequality in Eq. 6, then

$$\begin{aligned}\mathsf{QHT}_{\alpha}(\mathsf{U},|w\rangle) &= \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{m} |w_j|^2 \left(\frac{4}{\alpha_j^2}\right)} \le \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{m} |w_j|^2 \left(1+\cot^2\left(\frac{\alpha_j}{2}\right)\right)} \\ &= \frac{1}{\sqrt{2}}\sqrt{1+\frac{1}{2}\mathsf{QHT}^2_{\cot}(\mathsf{U},|w\rangle)}.\end{aligned}$$

We obtain the second inequality in the lemma by applying the inequality $\sqrt{a+b} \le \sqrt{a}+\sqrt{b}$, for positive reals a and b. ◀

Approximating Language Edit Distance Beyond Fast Matrix Multiplication: Ultralinear Grammars Are Where Parsing Becomes Hard!*

Rajesh Jayaram[1] and Barna Saha†[2]

1 **Brown University, Providence, RI, USA**
rajesh_jayaram@brown.edu
2 **University of Massachusetts Amherst, Amherst, MA, USA**
barna@cs.umass.edu

Abstract

In 1975, a breakthrough result of L. Valiant showed that parsing context free grammars can be reduced to Boolean matrix multiplication, resulting in a running time of $O(n^\omega)$ for parsing where $\omega \leq 2.373$ is the exponent of fast matrix multiplication, and n is the string length. Recently, Abboud, Backurs and V. Williams (FOCS 2015) demonstrated that this is likely optimal; moreover, a combinatorial $o(n^3)$ algorithm is unlikely to exist for the general parsing problem[1]. The language edit distance problem is a significant generalization of the parsing problem, which computes the minimum edit distance of a given string (using insertions, deletions, and substitutions) to any valid string in the language, and has received significant attention both in theory and practice since the seminal work of Aho and Peterson in 1972. Clearly, the lower bound for parsing rules out any algorithm running in $o(n^\omega)$ time that can return a nontrivial multiplicative approximation of the language edit distance problem. Furthermore, combinatorial algorithms with cubic running time or algorithms that use fast matrix multiplication are often not desirable in practice.

To break this n^ω hardness barrier, in this paper we study *additive* approximation algorithms for language edit distance. We provide two explicit combinatorial algorithms to obtain a string with minimum edit distance with performance dependencies on either the number of *non-linear productions*, k^*, or the number of *nested non-linear production*, k, used in the optimal derivation. Explicitly, we give an additive $O(k^*\gamma)$ approximation in time $O(|G|(n^2 + \frac{n^3}{\gamma^3}))$ and an additive $O(k\gamma)$ approximation in time $O(|G|(n^2 + \frac{n^3}{\gamma^2}))$, where $|G|$ is the grammar size and n is the string length. In particular, we obtain tight approximations for an important subclass of context free grammars known as *ultralinear* grammars, for which k and k^* are naturally bounded. Interestingly, we show that the same conditional lower bound for parsing context free grammars holds for the class of ultralinear grammars as well, clearly marking the boundary where parsing becomes hard!

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Approximation, Edit Distance, Dynamic Programming, Context Free Grammar, Hardness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.19

* A full version of the paper is available at https://web.cs.umass.edu/publication/docs/2017/UM-CS-2017-008.pdf.

† Research supported by NSF CRII 1464310, NSF CAREER 1652303, a Google Faculty Research Award, and a Yahoo ACE Award.

[1] with any polynomial dependency on the grammar size

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl; Article No. 19; pp. 19:1–19:15

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

1 Introduction

Introduced by Chomsky in 1956 [11], context-free grammars (CFG) play a fundamental role in the development of formal language theory [2, 22], compiler optimization [16, 42], natural language processing [27, 31], with diverse applications in areas such as computational biology [39], machine learning [33, 20, 41] and databases [23, 14, 34]. Parsing CFG is a basic computer science question, that given a CFG G over an alphabet Σ, and a string $x \in \Sigma^*$, $|x| = n$, determines if x belongs to the language $\mathcal{L}(G)$ generated by G. The canonical parsing algorithms such as Cocke-Younger-Kasimi (CYK) [2], Earley parser, [12] etc. are based on a natural dynamic programming, and run in $O(n^3)$ time[2]. In 1975, in a theoretical breakthrough, Valiant gave a reduction from parsing to Boolean matrix multiplication, showing that the parsing problem can be solved in $O(n^\omega)$ time [38]. Despite decades of efforts, these running times have remain completely unchanged.

Nearly three decades after Valiant's result, Lee came up with an ingenious reduction from Boolean matrix multiplication to CFG parsing, showing for the first time why known parsing algorithms may be optimal [25]. A remarkable recent result of Abboud, Backurs and V. Williams made her claims concrete [1]. Based on a conjecture of the hardness of computing large cliques in graphs, they ruled out any improvement beyond Valiant's algorithm; moreover they showed that there can be no combinatorial algorithm for CFG parsing that runs in truly subcubic $O(n^{3-\epsilon})$ time for $\epsilon > 0$ [1]. However combinatorial algorithms with cubic running time or algorithms that use fast matrix multiplication are often impractical. Therefore, a long-line of research in the parsing community has been to discover subclasses of context free grammars that are sufficiently expressive yet admit efficient parsing time [26, 24, 17]. Unfortunately, there still exist important subclasses of the CFG's for which neither better parsing algorithms are known, nor have conditional lower bounds been proven to rule out the possibility of such algorithms.

Language Edit Distance

A generalization of CFG parsing, introduced by Aho and Peterson in 1972 [3], is *language edit distance* (LED) which can be defined as follows.

▶ **Definition 1** (Language Edit Distance (LED))**.** Given a formal language $\mathcal{L}(G)$ generated by a grammar G over alphabet Σ, and a string $\overline{x} \in \Sigma^*$, compute the minimum number of edits (insertion, deletion and substitution) needed on $\overline{x}$ to convert it to a valid string in $\mathcal{L}(G)$.

LED is among the most fundamental and best studied problems related to strings and grammars [3, 30, 34, 35, 8, 1, 32, 6, 21], and generalizes two basic problems in computer science: parsing and string edit distance computation. Aho and Peterson presented a dynamic programming algorithm for LED that runs in $O(|G|^2 n^3)$ time [3], which was improved to $O(|G|n^3)$ by Myers in 1985 [30]. Only recently these bounds have been improved by Bringmann, Grandoni, Saha, and V. Williams to give the first truly subcubic $O(n^{2.8244})$ algorithm for LED [8]. When considering approximate answers, a *multiplicative* $(1+\epsilon)$-approximation for LED has been presented by Saha in [35], that runs in $O(\frac{n^\omega}{\text{poly}(\epsilon)})$ time.

These subcubic algorithms for LED crucially use fast matrix multiplication, and hence are not practical. Due to the hardness of parsing [25, 1], LED cannot be approximated

[2] Dependency on the grammar size if not specified is either $|G|$ as in most combinatorial algorithms, or $|G|^2$ as in most algebraic algorithms. In this paper the algorithms will depend on $|P|$, the number of productions in the grammar. In general we assume $|P| \in \Theta(|G|)$.

with any multiplicative factor in time $o(n^\omega)$. Moreover, there cannot be any combinatorial multiplicative approximation algorithm that runs in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$ [1]. LED provides a very generic framework for modeling problems with vast applications [23, 20, 41, 28, 33, 31, 15]. A fast exact or approximate algorithm for it is likely to have tangible impact, yet there seems to be a bottleneck in improving the running time beyond $O(n^\omega)$, or even in designing a truly subcubic combinatorial approximation algorithm. Can we break this n^ω barrier?

One possible approach is to allow for an *additive approximation*. Since the hardness of multiplicative approximation arise from the lower bound of parsing, it is possible to break the n^ω barrier by designing a purely combinatorial algorithm for LED with an additive approximation. Such a result will have immense theoretical and practical significance. Due to the close connection of LED with matrix products, all-pairs shortest paths and other graph algorithms [35, 8], this may imply new algorithms for many other fundamental problems. In this paper, we make a significant progress in this direction by providing the first nontrivial additive approximation for LED that runs in quadratic time. Let $G = (Q, \Sigma, P, S)$ denote a context free grammar, where Q is the set of nonterminals, Σ is the alphabet or set of terminals, P is the set of productions, and S is the starting non-terminal.

▶ **Definition 2.** Given $G = (Q, \Sigma, P, S)$, a production $A \to \alpha$ is said to be *linear* if there is at most one non-terminal in α where $A \in Q$ and $\alpha \in (Q \cup \Sigma)^*$. Otherwise, if α contains two or more non-terminals, then $A \to \alpha$ is said to be *non-linear*.

The performance of our algorithms depends on either the total number of *non-linear productions* or the maximum number of *nested non-linear productions* (depth of the parse tree after condensing every consecutive sequence of linear productions, see the full version for more details) in the derivation of string with optimal edit distance, where the latter is often substantially smaller. Explicitly, we give an additive $O(k^*\gamma)$ approximation in time $O(|G|(n^2 + \frac{n^3}{\gamma^3}))$ and an additive $O(k\gamma)$ approximation in time $O(|G|(n^2 + \frac{n^3}{\gamma^2}))$, where k^* is the number of non-linear productions in the derivation of the optimal string, and k is the maximum number of nested non-linear productions in the derivation of the optimal string (each minimized over all possible derivations). Our algorithms will be particularly useful for an important subclass of CFGs, known as the *ultralinear grammars*, for which these values are tightly bounded for all derivations [43, 10, 26, 7, 29].

▶ **Definition 3** (ultralinear)**.** A grammar $G = (Q, \Sigma, P, S)$ is said to be **k-ultralinear** if there is a partition $Q = Q_1 \cup Q_1 \cup \cdots \cup Q_k$ such that for every $X \in Q_i$, the productions of X consist of *linear productions* $X \to \alpha A | A\alpha | \alpha$ for $A \in Q_j$ with $j \leq i$ and $\alpha \in \Sigma$, or *non-linear productions* of the form $X \to w$, where $w \in (Q_1 \cup Q_2 \cup \cdots \cup Q_{i-1})^*$.

The parameter k places a built-in upper bound on the number of nested non-linear productions allowed in any derivation. Thus for simplicity we will use k both to refer to the parameter of an ultralinear grammar, as well as the maximum number of nested non-linear productions. Furthermore, if d is the maximum number of non-terminals on the RHS of a production, then d^k is a built-in upper bound on the total number of non-linear productions in any derivation. In all our algorithms, without loss of generality, we use a standard normal form where $d = 2$ for all non-linear productions. As we will see later, given any CFG G and any $k \geq 1$, we can create a new grammar G' by making k copies $Q_1, \ldots, Q_k$ of the set of non-terminals Q of G, and forcing every nonlinear production in Q_i to go to non-terminals in Q_{i-1}. Thus G' has non-terminal set $Q_1 \cup Q_2 \cup \cdots \cup Q_k$, and size $O(k|G|)$. In this way we can restrict any CFG to a k-ultralinear grammar which can produce any string in $\mathcal{L}(G)$

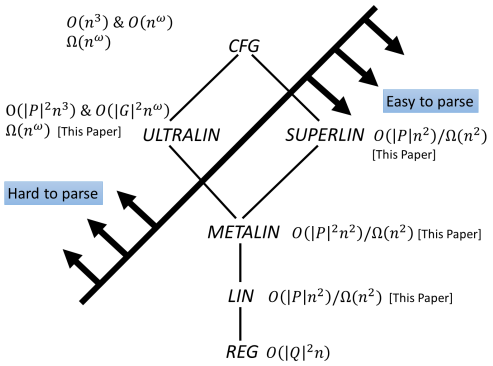


Figure 1 CFG Hierarchy: Upper bounds shown first followed by lower bounds for each class of grammars. Here $|P|$ is the number of productions in the grammar [38] [1] [40].

requiring no more than k nested non-linear productions. It is precisely this procedure of creating a k-ultralinear grammar from a CFG G that we use in our proof of hardness for parsing ultralinear languages (see the full version).

For example, if G is the well-known *Dyck Languages* [34, 6], the language of well-balanced parenthesis, $\mathcal{L}(G')$ contains the set of all parentheses strings with at most k levels of nesting. Note that a string consisting of n open parenthesis followed by n matching closed parenthesis has zero levels of nesting, whereas the string "((())())" has one level. As an another example, consider RNA-folding [8, 39, 44] which is a basic problem in computational biology and can be modeled by grammars. The restricted language $\mathcal{L}(G')$ for RNA-folding denotes the set of all RNA strings with at most k nested folds. In typical applications, we do not expect the number of nested non-linear productions used in the derivation of a valid string to be too large [14, 23, 4].

Among our other results, we consider exact algorithms for several other notable subclasses of the CFG's. In particular, we develop exact quadratic time language edit distance algorithms for the linear, metalinear, and superlinear languages. Moreover, we show matching lower bound assuming the Strong Exponential Time Hypothesis [18, 19]. The figure to the right displays the hierarchical relationship between these grammars, where all upwards lines denote strict containment. Interestingly, till date there exists no parsing algorithm for the ultralinear grammars that runs in time $o(n^\omega)$, while a $O(n^2)$ algorithm exists for the metalinear grammars. In addition, there is no combinatorial algorithm that runs in $o(n^3)$ time. In this paper, we derive conditional lower bound exhibiting why a faster algorithm has so far been elusive for the ultralinear grammars, clearly demarking the boundary where parsing becomes hard!

1.1 Results & Techniques

Lower Bounds. Our first hardness result is a lower bound for the problem of linear language edit distance. We show that a truly subquadratic time algorithm for linear language edit distance would refute the Strong Exponential Time Hypothesis (SETH). This further builds on a growing family of "SETH-hard" problems – those for which lower bounds can be proven conditioned on SETH. We prove this result by reducing binary string edit distance, which has been shown to be SETH-hard [9, 5], to linear language edit distance.

▶ **Theorem** (Linear Grammar Hardness of Parsing). *There exists no algorithm to compute the minimum edit distance between a string $\overline{x}$, $|\overline{x}| = n$, and a linear language $\mathcal{L}(G)$ in $o(n^{2-\epsilon})$ time for any constant $\epsilon > 0$, unless SETH is false.*

Our second, and primary hardness contribution is a conditional lower bound on the recognition problem for ulralinear languages. Our result builds closely off of the work of Abboud, Backurs and V. Williams [1], who demonstrate that finding an $o(n^3)$-time combinatorial algorithm or any $o(n^\omega)$-algorithm for context free language recognition would result in faster algorithms for the k-clique problem and falsify a well-known conjecture in graph algorithms. We modify the grammar in their construction to be ultralinear, and then demonstrate that the same hardness result holds for our grammar. See the full version for details.

▶ **Theorem** (Ultralinear Grammar Hardness of Parsing). *There is a ultralinear grammar $\mathcal{G}_U$ such that if we can solve the membership problem for a string of length n in time $O(|\mathcal{G}_U|^\alpha n^c)$ for any fixed constant $\alpha > 0$, then we can solve the $3k$-clique problem on a graph with n nodes in time $O(n^{c(k+3)+3\alpha})$.*

Upper Bounds. We provide the first quadratic time algorithms for linear (Theorem 7), superlinear (in full version), and metalinear language edit distance (in full version), running in $O(|P|n^2)$, $O(|P|n^2)$ and $O(|P|^2n^2)$ time respectively. This exhibits a large family of grammars for which edit distance computation can be done faster than for general context free grammars, as well as for other well known grammars such as the Dyck grammar [1]. Along with our lower bound for the ultralinear language parsing, this demonstrates a clear division between those grammars for which edit distance can be efficiently calculated, and those for which the problem is likely to be fundamentally hard. Our algorithms build progressively off the construction of a *linear language edit distance graph*, reducing the problem of edit distance computation to computing shortest path on a graph with $O(|P|n^2)$ edges (Section 2).

Our main contribution is an additive approximation for language edit distance. We first present a cubic time exact algorithm, and then show a general procedure for modifying this algorithm, equivalent to forgetting states of the underlying dynamic programming table, into a family of *amnesic* dynamic programming algorithms. This produces *additive approximations* of the edit distance, and also provides a tool for proving general bounds on any such algorithm. In particular, we provide two explicit procedures for forgetting dynamic programming states: *uniform* and *non-uniform* grid approximations achieving the following approximation-running time trade-off. See Section 4, and the full version for missing proofs.

▶ **Theorem 4.** *If $\mathcal{A}$ is a γ-uniform grid approximation, then the edit distance computed by $\mathcal{A}$ satisfies $|OPT| \leq |\mathcal{A}| \leq |OPT| + O(k^*\gamma)$ and it runs in $O(|P|(n^2 + (\frac{n}{\gamma})^3))$ time.*

▶ **Theorem 5.** *Let $\mathcal{A}$ be any γ-non-uniform grid approximation, then the edit distance computed by $\mathcal{A}$ satisfies $|OPT| \leq |\mathcal{A}| \leq |OPT| + O(k\gamma)$and it runs in $O\big(|P|\big(n^2 + \frac{n^3}{\gamma^2}\big)\big)$ time.*

We believe that our amnesic technique can be applied to wide range of potential dynamic programming approximate algorithms, and lends itself particularly well to randomization.

2 Linear Grammar Edit Distance in Quadratic Time

We first introduce a graph-based exact algorithm for linear grammar, that is a grammar $G = (Q, \Sigma, P, S)$ where every production has one of the following forms: $A \to \alpha B$, $A \to B\alpha$, $A \to \alpha B\beta$, or $A \to \alpha$ where $A, B \in Q$, and $\alpha, \beta \in \Sigma$. Given G and a string $\overline{x} = x_1x_2 \dots x_n \in \Sigma^*$, we give an $O(|P|n^2)$ algorithm to compute edit distance between $\overline{x}$ and G in this section. The algorithm serves as a building block for the rest of the paper.

Note that if we only have productions of the form $A \to \alpha B$ (or $A \to B\alpha$ but not both) then the corresponding language is regular, and all regular languages can be generated in

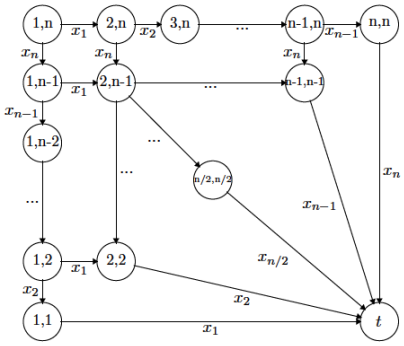


Figure 2 Clouds corresponding to Linear Grammar Edit Distance Graph Construction. Each cloud contains a vertex for every nonterminal.

this manner. However, there are linear languages that are not regular. For instance, the language $\{0^n1^n \mid n \in \mathbb{N}\}$ can be produced by the linear grammar $S \to 0S1 \mid \epsilon$, but cannot be produced by any regular grammar [37]. Therefore, regular languages are a strict subclass of linear languages. Being a natural extension of the regular languages, the properties and applications of linear languages are of much interest[13, 36].

Algorithm. Given inputs G and $\overline{x}$, we construct a weighted digraph $\mathcal{T} = \mathcal{T}(G, \overline{x})$ with a designated vertex $S^{1,n}$ as the source and t as the sink such that the weight of the shortest path between them will be the minimum language edit distance of $\overline{x}$ to G.

Construction. The vertices of $\mathcal{T}$ consist of $\binom{n}{2}$ *clouds*, each corresponding to a unique substring of $\overline{x}$. We use the notation (i, j) to represent the cloud, $1 \leq i \leq j \leq n$, corresponding to the substring $x_ix_{i+1}....x_j$. Each cloud will contain a vertex for every nonterminal in Q. Label the nonterminals $Q = \{S = A_1, A_2, \ldots, A_q\}$ where $|Q| = q$, then we denote the vertex corresponding to A_k in cloud (i, j) by $A_k^{i,j}$. We will add a new sink node t, and use $S^{1,n}$ as the source node s. Thus the vertex set of $\mathcal{T}$ is $V(\mathcal{T}) = \{A_k^{i,j} \mid 1 \leq i \leq j \leq n,\ 1 \leq k \leq q\} \cup \{t\}$. The edges of $\mathcal{T}$ will correspond to the productions in G. Each path from a nonterminal $A_k^{i,j}$ in (i, j) to t corresponds to the production of a legal string w, that is a string that can be derived starting from A_k and following the productions of P, and a sequence of editing procedures to edit w to $x_ix_{i+1} \ldots x_j$. For any cloud (i, j), edges will exist between two nonterminals in (i, j), and from nonterminals in (i, j) to nonterminals in $(i + 1, j)$ and $(i, j - 1)$. Our goal will be to find the shortest path from $S^{1,n}$, the starting nonterminal S in cloud $(1, n)$, to the sink t.

Adding the edges. Each edge in $\mathcal{T}$ is directed, has a weight in $\mathbb{Z}^+$ and a label from $\{x_1, x_2, .., x_n, \epsilon\} \cup \{\epsilon(\alpha) \mid \alpha \in \Sigma\}$, where $\epsilon(\alpha)$ corresponds to the deletion of α. If u, v are two vertices in $\mathcal{T}$, then we use the notation $u \xrightarrow[w(u,v)]{\ell} v$ to denote the existence of an edge from u to v with weight $w(u, v)$ and edge label ℓ. For any nonterminal $A \in Q$, define $null(A)$ to be the length of the shortest string in Σ^* derivable from A, which can be precomputed in $O(|Q||P| \log(|Q|))$ time for all $A \in Q$ (see full version for details). This is the minimum cost of deleting a whole string produced by A. Given input $x_1x_2 \ldots x_n$, for all nonterminals A_k, A_r and every $1 \leq i \leq j \leq n$, the construction is as follows:

- **Legal Productions:** For $i \neq j$, then if $A_k \to x_i A_r$ is a production, add the edge $A_k^{i,j} \xrightarrow[0]{x_i} A_r^{i+1,j}$ to $\mathcal{T}$. If $A_k \to A_r x_j$ is a production, add the edge $A_k^{i,j} \xrightarrow[0]{x_j} A_r^{i,j-1}$ to $\mathcal{T}$.
- **Completing Productions:** If $A_k \to x_i$ is a production, add the edge $A_k^{i,i} \xrightarrow[0]{x_i} t$ to $\mathcal{T}$. If $A_k \to x_i A_r$ or $A_k \to A_r x_i$ is a production, add the edge $A_k^{i,i} \xrightarrow[null(A_r)]{x_i} t$ to $\mathcal{T}$.
- **Insertion:** If $A_k \to x_i A_k$ is *not* a production, add the edge $A_k^{i,j} \xrightarrow[1]{x_i} A_k^{i+1,j}$ to $\mathcal{T}$. If $A_k \to A_k x_j$ is *not* a production, add $A_k^{i,j} \xrightarrow[1]{x_j} A_k^{i,j-1}$. *{these are called insertion edges.}*
- **Deletion:** For every production $A_k \to \alpha A_r$ or $A_k \to A_r \alpha$, add the edge $A_k^{i,j} \xrightarrow[1]{\epsilon(\alpha)} A_r^{i,j}$. *{these are called deletion edges.}*
- **Replacement:** For every production $A_k \to \alpha A_r$, if $\alpha \neq x_i$, then add the edge $A_k^{i,j} \xrightarrow[1]{x_i} A_r^{i+1,j}$ to T. For every production $A_k \to A_r \alpha$, if $\alpha \neq x_j$, add $A_k^{i,j} \xrightarrow[1]{x_j} A_r^{i,j-1}$ to $\mathcal{T}$. For any A_k such that $A_k \to x_i$ is not a production, but $A_k \to \alpha$ is a production with $\alpha \in \Sigma$, add the edge $A_k^{i,i} \xrightarrow[1]{x_i} t$ to $\mathcal{T}$.*{these are called substitution or replacement edges.}*

▶ **Theorem 6.** *For every $A_k \in Q$ and every $1 \leq i \leq j \leq n$, the cost of the shortest path of from $A_k^{i,j}$ to the sink $t \in \mathcal{T}$ is d if and only if d is the minimum edit distance between the string $x_i \dots x_j$ and the set of strings which can be derived from A_k.*

▶ **Theorem 7.** *The cost of the shortest path from $S^{1,n}$ to t in the graph $\mathcal{T}$ is the minimum edit distance which can be computed in $O(|P|n^2)$ time.*

3 Context Free Language Edit Distance

In this section, we develop an exact algorithm which utilizes the graph construction presented in Section 2 to compute the language edit distance of a string $\overline{x} = x_1 \dots x_n$ to any context free grammar (CFG) $G = (Q, \Sigma, P, S)$. We use a standard normal form for G, which is Chomsky normal form except we also allow productions of the form $A \to Aa|aA$, where $A \in Q, a \in \Sigma$. For us, the important property of this normal form is that every non-linear production must be of the form $A \to BC$, with exactly two non-terminals on the right hand side. Any CFG can be reduced to this normal form (see full version for more details).

Let $P_L, P_{NL} \subset P$ be the subsets of (legal) linear and non-linear productions respectively. Then for any nonterminal $A \in Q$, the grammar $G_L = (Q, \Sigma, P_L, A)$ is linear, and we denote the corresponding linear language edit distance graph by $\mathcal{T}(G_L, \overline{x}) = \mathcal{T}$, as constructed in Section 2. Let L_i be the set of clouds in $\mathcal{T}$ which correspond to substrings of length i (so $L_i = \{(k, j) \in \mathcal{T} \mid j - k + 1 = i\}$). Then $L_1, \dots, L_n$ is a *layered partition* of $\mathcal{T}$. Let t be the sink of $\mathcal{T}$. We write $\mathcal{T}^R$ to denote the graph $\mathcal{T}$ where the direction of each edge is reversed. Let L_i^R denote the edge reversed subgraph of L_i. In other words, L_i^R is the subgraph of $\mathcal{T}^R$ with the same vertex set as L_i. Our algorithm will add some additional edges within L_i^R, and some additional edges from t to L_i^R, for all $1 \leq i \leq n$, resulting in an augmented subgraph which we denote $\overline{L}_i^R$. We then compute single source shortest path from t to $\overline{L}_i^R \cup \{t\}$ in phase i. Our algorithm will maintain the property that, after phase $q - p + 1$, if $A^{p,q}$ is any nonterminal in cloud (p, q) then the weight of the shortest path from t to $A^{p,q}$ is precisely the minimum edit distance between the string $x_p x_{p+1} \dots x_q$ and the set of strings that are legally derivable from A. The algorithm is as follows:

Algorithm: Context Free-Exact

1. **Base Case: strings of length 1.** For every non-linear production $A \to BC$, and every $1 \leq \ell \leq n$, add the edges $A^{\ell,\ell} \xleftarrow[null(B)]{} C^{\ell,\ell}$ and $A^{\ell,\ell} \xleftarrow[null(C)]{} B^{\ell,\ell}$ to L_1^R. Note that the direction of the edges are reversed because we are adding edges to L_1^R and not L_1. Call the resulting augmented graph $\overline{L_1^R}$.
2. Solve single source shortest path from t to every vertex in $\overline{L_1^R} \cup \{t\}$. Store the value of the shortest path from t to every vertex in $\overline{L_1^R}$, and an encoding of the path itself. For any $1 \leq p \leq q \leq n$ and $A^{p,q} \in L_{q-p+1}$, we write $T_{p,q}(A)$ to denote the weight of the shortest path from t to $A^{p,q}$. After having computed shortest paths from t to every vertex in the subgraphs $\overline{L}_1^R, \ldots, \overline{L}_{i-1}^R$, we now consider L_i^R.
3. **Induction: strings of length i.** For every edge from a vertex $A^{p,q}$ in L_i to a vertex $B^{p+1,q}$ or $B^{p,q-1}$ in L_{i-1} with cost $\gamma \in \{0,1\}$, add an edge from t to $A^{p,q} \in L_i^R$ with cost $T_{p+1,q}(B) + \gamma$ or $T_{p,q-1}(B) + \gamma$, respectively. These are the linear production edges created in the linear grammar edit distance algorithm.
4. For every non-linear production $A \to BC$ and every vertex $A^{p,q} \in L_i^R$, add an edge from t to $A^{p,q}$ in L_i^R with cost c where $c = \min_{p \leq \ell < q} T_{p,\ell}(B) + T_{\ell+1,q}(C)$. The indices $p \leq \ell < q$ are called *splitting points,* as they specify where the string $x_p, \ldots, x_q$ is split by the production $A \to BC$. To later recover the derivation, we store the specific ℓ which yields the minimum value of the above equation.
5. For every non-linear production $A \to BC$, add the edge $A^{p,q} \xleftarrow[null(B)]{} C^{p,q}$ and $A^{p,q} \xleftarrow[null(C)]{} B^{p,q}$ to L_i^R.
6. After adding the edges in steps 3-5, we call the resulting graph $\overline{L}_i^R$. Then compute shortest path from t to every vertex in the subgraph $\overline{L}_i^R \cup \{t\}$, and store the values of the shortest paths, along with an encoding of the paths themselves.
7. Repeat for $i = 1, 2, \ldots, n$. Return the value $T_{1,n}(S)$.

▶ **Theorem 8.** *For any nonterminal $A \in Q$ and $1 \leq p \leq q \leq n$, the weight of the shortest path from $A^{p,q} \in \overline{L}_i$ to t is the minimum edit distance between the substring $x_p \ldots x_q$ and the set of strings which can be legally produced from A, and the overall time required to compute the language edit distance is $O(|P|n^3)$.*

4 Context Free Language Edit Distance Approximation

Now this cubic time algorithm itself is not an improvement on that of Aho and Peterson [3]. However, by strategically modifying the construction of the subgraphs L_i by **forgetting** to compute some of the non-linear edge weights (and taking the minimum over fewer splitting points for those that we do compute), we can obtain an additive approximation of the minimum edit distance. We introduce a family of approximation algorithms which do just this, and prove a strong general bound on their behavior. Our results give bounds for the performance of our algorithm for any CFG. Additionally, for any k-ultralinear language, our results also give explicit $O(k\sqrt{n})$ and $O(2^k n^{1/3})$ additive approximations from this family which run in quadratic time. Note that, as shown in our construction in the proof of hardness of parsing ultralinear grammars, for any k we can restrict any context free grammar G to a k-ultralinear grammar G' such that $\mathcal{L}(G') \subseteq \mathcal{L}(G)$ contains all words that can be derived using fewer than $\leq k$ nested non-linear productions (see full version for a more formal definition of k and hardness proofs).

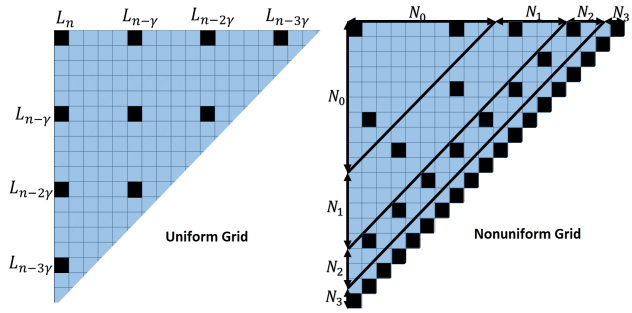


Figure 3 Non-uniform edges are computed only for a subset of the clouds (colored black). Only a subset of the splitting points are considered while computing the weights.

▶ **Definition 9.** For any Context Free Language edit distance approximation algorithm $\mathcal{A}$, we say that $\mathcal{A}$ is in the family $\mathcal{F}$ if it follows the same procedure as in the exact algorithm with the following modifications:

1. **Subset of non-linear productions.** $\mathcal{A}$ constructs the non-linear production edges in step 4 for the vertices in some subset of the total set of clouds $\{(p,q) \mid 1 \leq p \leq q \leq n\}$.
2. **Subset of splitting points.** For every cloud (p,q) that $\mathcal{A}$ computes non-linear production edges for, in step 4 of the algorithm when computing the weight c of any edge in this cloud it takes minimum over only a subset of the possible splitting points $p, \ldots, q$ (where this subset is the same for every non-linear edge weight computed in (p,q)).

By forgetting to construct all non-linear production edges, and by taking a minimum over fewer values when we do construct non-linear production edges, the time taken by our algorithm to construct new edges can be substantially reduced. Roughly, the intuition for how we can still obtain an additive approximation is as follows. If the shortest path to the sink in the exact algorithm uses a non-linear edge from a vertex $A^{p,q}$ in cloud (p,q), then naturally our approximation algorithm would also use such an edge if it existed. However, it is possible that nonlinear edges were not constructed for cloud (p,q) by the approximation. Still, what we can do is find the closest cloud (p',q'), with $p \leq p' \leq q' \leq q$, such that nonlinear edges were constructed in (p',q'), and then follow the insertion edges $A^{p,q} \to A^{p+1,q} \to \cdots \to A^{p',q'}$, and take the desired non-linear production edge from $A^{p',q'}$. The additional incurred cost is at most $|p-p'| + |q-q'|$, or the distance to the nearest cloud with non-linear edges, and this cost is incurred at most once for every non-linear production in an optimal derivation.

We now give two explicit examples of how steps 1 and 2 can be implemented. We later prove explicit bounds on the approximations of these examples in Theorems 4 and 5. In both examples a *sensitivity parameter*, γ, is first chosen. We use $|OPT|$ to denote the optimum language edit distance, and $|\mathcal{A}|$ to denote the edit distance computed by an approximation algorithm $\mathcal{A}$.

▶ **Definition 10.** An approximation algorithm $\mathcal{A} \in \mathcal{F}$ is a γ-*uniform grid* approximation if for $i = n, (n-\gamma), (n-2\gamma), \ldots, (n - \lfloor \frac{n}{\gamma} \rfloor \gamma)$ (see Figure 3).

1. $\mathcal{A}$ constructs non-linear production edges only for an evenly-spaced $1/\gamma$ fraction of the clouds in L_i, and no others, where γ is a specified sensitivity parameter.
2. Furthermore, for every non-linear edge constructed, $\mathcal{A}$ considers only an evenly-spaced $1/\gamma$ fraction of the possible break points.

Here if i or $(n-i+1)$ (the number of substrings of length i) is not evenly divisible by γ, we evenly space the clouds/breakpoints until no more will fit.

We will later see that the running time of such a γ-uniform grid approximation is $O(|P|(n^2 + (\frac{n}{\gamma})^3))$, and in particular for any k-ultralinear grammar G it gives an additive approximation of $O(2^k\gamma)$. Thus by setting $\gamma = n^{1/3}$, we get an $O(2^k n^{1/3})$-approximation in $O(|P|n^2)$ time (Theorem 4).

▶ **Definition 11.** For $i = 0, 1, \ldots, \log(n)$, set $N_i = \{L_j \mid \frac{n}{2^{i+1}} < j \leq \frac{n}{2^i}\}$. Let $N_i' \subset N_i$ be an evenly-spaced $\min\{\frac{2^i}{\gamma}, 1\}$-fraction of the L_j's in N_i (subset of diagonals). Then, an approximation algorithm $\mathcal{A} \in \mathcal{F}$ is a γ-*non-uniform grid* approximation if, for every $L_j \in N_i'$, $\mathcal{A}$ computes non-linear production edges only for a $\min\{\frac{2^i}{\gamma}, 1\}$-evenly-spaced fraction of the clouds in L_j. Furthermore, for any of these clouds in N_i' for which $\mathcal{A}$ does compute non-linear production edges, $\mathcal{A}$ considers only an evenly-spaced $\min\{\frac{2^i}{\gamma}, 1\}$ -fraction of all possible break points (see Figure 3 (right)).

We will see that the running time of a γ-non-uniform grid approximation is $O(|P|(n^2 + \frac{n^3}{\gamma^2}))$, and in particular for any k-ultralinear grammar, or if k is the maximum number of nested non-linear productions, it gives an additive approximation of $O(k\gamma)$. Hence setting $\gamma = \sqrt{n}$, we get an additive approximation of $O(k\sqrt{n})$ in quadratic time (Theorem 5).

4.1 Analysis

The rest of this section will be devoted to proving bounds on the performance of approximation algorithms in $\mathcal{F}$. We use $\mathcal{T}^{OPT}$ to denote the graph which results from adding all the edges specified in the exact algorithm to $\mathcal{T}$. Recall that $\mathcal{T}$ is the graph constructed from the linear productions in G. For $\mathcal{A} \in \mathcal{F}$, we write $\mathcal{T}^{\mathcal{A}}$ to denote the graph which results from adding the edges specified by the approximation algorithm $\mathcal{A}$. Note that since $\mathcal{A}$ functions by forgetting to construct a subset of the non-linear edges created by the exact algorithm, we have that the edge sets satisfy $E(\mathcal{T}) \subseteq E(\mathcal{T}^{\mathcal{A}}) \subseteq E(\mathcal{T}^{OPT})$. We now introduce the primary structure which will allow us to analyze the execution of our language edit distance algorithms.

Binary Production-Edit Trees

▶ **Definition 12.** A *production-edit tree* (PET) $\mathbb{T}$ for grammar G and input string $\overline{x}$ is a binary tree which satisfies the following properties:

1. Each node of $\mathbb{T}$ stores a path in the linear grammar edit distance graph $\mathcal{T} = \mathcal{T}(G_L, \overline{x})$ (see Section 2 and 3). The path given by the root of $\mathbb{T}$ must start at the source vertex $S^{1,n}$ of $\mathcal{T}$.
2. For any node $v \in \mathbb{T}$, let $A^{p,q}, B^{r,s}$ be the starting and ending vertices of the corresponding path. If $B^{r,s}$ is not the sink t of $\mathcal{T}$, then v must have two children, v_R, v_L, such that there exists a production $B \to CD$ and the starting vertices of the paths in v_L and v_R are $C^{r,\ell}$ and $D^{\ell+1,s}$ respectively, where ℓ is some splitting point $r - 1 \leq \ell \leq s$. If $\ell = r - 1$ or $\ell = s$, then one of the children will be in the same cloud (r, s) as the ending cloud of the path given by v, and the other will be called a *nullified node*. This corresponds to the case where one of the *null* edges created in step 5 of the exact algorithm is taken.
3. If the path in $v \in \mathbb{T}$ ends at the sink of $\mathcal{T}$, then v must be a leaf in $\mathbb{T}$. If $A^{p,q}$ is the starting vertex of the path, this means that the path derives the entire substring $x_p \ldots x_q$ using only linear productions. Thus a node v is a leaf of $\mathbb{T}$ if and only if it either ends at the sink or is a nullified node. It follows from 2. and 3. that every non-leaf node has exactly 2 children.

Notation: To represent a node in $\mathbb{T}$ that is a path of cost c from $A^{p,q}$ to either $B^{r,s}$, or t, we will use the notation $[A^{p,q}, B^{r,s}, c]$, or $[A^{p,q}, t, c]$, respectively. If one of the arguments is either unknown or irrelevant, we write $\cdot$ as a placeholder. In the case of a nullified node, corresponding to the nullification of $A \in Q$, we write $[A, t, null(A)]$ to denote the node. Note, since we are now dealing with two *types* of graphs, to avoid confusion whenever we are talking about a vertex $A^{p,q}$ in any of the edit-distance graphs (such as $\mathcal{T}, \mathcal{T}^{\mathcal{A}}, \mathcal{T}^{OPT}$, ect), we will use the term *vertex*. When referring to the elements of a PET $\mathbb{T}$ we will use the term *node*. Also note that all error productions are linear.

We can now represent any sequence of edits produced by a language edit distance algorithm by such a PET, where the edit distance is given by the sum of the costs stored in the nodes of the tree. To be precise, if $[\cdot, \cdot, c_1], \ldots, [\cdot, \cdot, c_k]$ is the set of all nodes in $\mathbb{T}$, then the associated total cost $\|\mathbb{T}\| = \sum_{i=1}^{k} c_i$. Let $\mathcal{D}_{\mathcal{A}}$ be the set of PET's $\mathbb{T}$ compatible with a fixed approximation algorithm $\mathcal{A} \in \mathcal{F}$.

▶ **Definition 13** (PET's compatible with $\mathcal{A}$)**.** For an approximation algorithm $\mathcal{A} \in \mathcal{F}$, let $\mathcal{D}_{\mathcal{A}} \subset \mathcal{F}$ be the set of PET's $\mathbb{T}$ which satisfy the following constraints:

1. If $[A^{p,q}, B^{r,s}, \cdot]$ is a node in $\mathbb{T}$, where $A, B \in Q$, then $\mathcal{A}$ must compute non-linear edges for the cloud $(r, s) \in \mathcal{T}^{\mathcal{A}}$.
2. If $[C^{r,\ell}, \cdot, \cdot], [D^{\ell+1,s}, \cdot, \cdot]$ are the left and right children of a node $[A^{p,q}, B^{r,s}, \cdot]$ respectively, then $\mathcal{A}$ must consider the splitting point $\ell \in [p, q)$ when computing the weights of the non-linear edges in the cloud $(r, s) \in \mathcal{T}^{\mathcal{A}}$.

The set $\mathcal{D}_{\mathcal{A}}$ is then the set of all PET's which utilize only the non-linear productions and splitting points which correspond to edges that are actually constructed by the approximation algorithm $\mathcal{A}$ in $\mathcal{T}^{\mathcal{A}}$. Upon termination, any $\mathcal{A} \in \mathcal{F}$ will return the value $\|\mathbb{T}_{\mathcal{A}}\|$ where $\mathbb{T}_{\mathcal{A}} \in \mathcal{D}_{\mathcal{A}}$ is the tree corresponding to the shortest path from t to $S^{1,n}$ in $\mathcal{T}^{\mathcal{A}}$. The following theorem is not difficult to show.

▶ **Theorem 14.** *Fix any $\mathcal{A} \in \mathcal{F}$, and let c be the edit distance returned after running the approximation algorithm $\mathcal{A}$. Then if $\mathbb{T}$ is any PET in $\mathcal{D}_{\mathcal{A}}$, we have $c \leq \|\mathbb{T}\|$.*

Note that since the edges of $\mathcal{T}^{\mathcal{A}}$ are a subset of the edges of $\mathcal{T}^{OPT}$ considered by an exact algorithm OPT, we also have $c \geq \|\mathbb{T}_{OPT}\|$, where $\mathbb{T}_{OPT}$ is the PET given by the exact algorithm. To prove an upper bound on c, it then suffices to construct a explicit $\mathbb{T} \in \mathcal{D}_{\mathcal{A}}$, and put a bound on the size of $\|\mathbb{T}\|$. Thus, in the remainder of our analysis our goal will be to construct such a $\mathbb{T} \in \mathcal{D}_{\mathcal{A}}$. We now introduce our precision functions.

▶ **Definition 15** (Precision Functions)**.** For any cloud $(p, q) \in \mathcal{T}^{\mathcal{A}}$, let $\alpha(p, q)$ be any upper bound on the minimum distance $d^*((p, q), (r, s)) = (r - p) + (q - s)$ such that $p \leq r \leq s \leq q$ and $\mathcal{A}$ computes non-linear edge weights for the cloud (r, s) . Let $\beta(p, q)$ be an upper bound on the maximum distance between any two splitting points which are considered by $\mathcal{A}$ in the construction of the non-linear production edges originating in a cloud (r, s) such that $\mathcal{A}$ computes non-linear edge weights for (r, s) and $d^*((p, q), (r, s)) \leq \alpha(p, q)$. Furthermore, the precision functions must satisfy $\alpha(p, q) \geq \alpha(p', q')$ and $\beta(p, q) \geq \beta(p', q')$ whenever $(q - p) \geq (q' - p')$.

While the approximation algorithms presented in this paper are deterministic, the definitions of $\alpha(p, q)$ and $\beta(p, q)$ allow the remaining theorems to be easily adapted to algorithms which *randomly forget* to compute non-linear edges. While our paper considers only two explicit approximation algorithms, stating our results in this full generality substantially

easies the analysis. Both Theorems 4 and 5 will follow easily once general bounds are proven, and without the generality two distinct proofs would be necessary.

Constructing a PET $\mathbb{T} \in \mathcal{D}_{\mathcal{A}}$ similar to $\mathbb{T}_{OPT}$

Our goal is now to construct a PET $\mathbb{T} \in \mathcal{D}_{\mathcal{A}}$ with bounded cost. We do this by considering each node v of $\mathbb{T}_{OPT}$ and constructing a corresponding node u in $\mathbb{T}$ such that the path stored in u *imitates* the path in v as closely as possible. A perfect imitation may not be feasible if the path at v uses a non-linear production edge in a cloud that $\mathcal{A}$ does not compute non-linear edges for. Whenever this happens, we will need to move to the closest cloud which $\mathcal{A}$ does consider before making the *same* non-linear production that the exact algorithm did. Afterwards, the ending cloud of our path will deviate from that of the optimal, so we will need to bound the total deviation that can occur throughout the construction of our tree in terms of $\alpha(p,q)$ and $\beta(p,q)$. The following lemma will be used crucially in this regard for the proof of our construction in Theorem 17. The lemma takes as input a node $[A^{p,q}, B^{r,s}, c] \in \mathbb{T}_{OPT}$ and a cloud (p',q') such that $x_p,\dots,x_q$ is not disjoint from $x_{p'},\dots,x_{q'}$, and constructs a path $[A^{p',q'}, B^{r',s'}, c']$ of bounded cost that is compatible with a PET $\mathbb{T} \in \mathcal{D}_{\mathcal{A}}$.

▶ **Lemma 16.** *Let $[A^{p,q}, B^{r,s}, c]$ be any non-leaf node in $\mathbb{T}_{OPT}$, and let $\mathcal{A} \in \mathcal{F}$ be an approximation algorithm with precision functions $\alpha(p,q), \beta(p,q)$. If p', q' satisfy $p \le q'$ and $p' \le q$, then there is a path from $A^{p',q'}$ to $B^{r',s'}$, where $r \le r' \le s' \le s$, of cost $c' \le c + (|p'-p| + |q'-q|) - (|r'-r| + |s'-s|) + 2\alpha(r,s)$ such that $\mathcal{A}$ computes non-linear production edges for cloud (r',s'). Furthermore, for any leaf node $[A^{p,q}, t, c] \in \mathbb{T}_{OPT}$, we can construct a path from $A^{p',q'}$ of cost at most $c' \le c + (|p'-p| + |q'-q|)$ to the sink.*

We will now iteratively apply Lemma 16 to each node $v \in \mathbb{T}_{OPT}$ from the root down, transforming it into a new node $\psi(v) \in \mathbb{T}$. Here ψ will be a surjective function $\psi : V(\mathbb{T}_{OPT}) \to V(\mathbb{T})$. Lemma 16 will guarantee that the cost of $\psi(v)$ is not too much greater than that of v. If during the construction of $\mathbb{T}$, the substrings corresponding to v and $\psi(v)$ become disjoint, then we will transform the *entire* subtree rooted at v into a single node $\psi(v) \in \mathbb{T}$, thus the function may not be injective.

Let v be any node in $\mathbb{T}_{OPT}$, and u its parent node if v is not the root. Let $(p,q),(r,s) \in \mathcal{T}$ and $(p_v,q_v),(r_v,s_v) \in \mathcal{T}$ be the starting and ending clouds of u and v respectively. Similarly let $(p',q'),(r',s')$ and $(p'_v,q'_v),(r'_v,s'_v)$ be the starting and ending clouds of $\psi(u)$ and $\psi(v)$ respectively. Furthermore, let (p_X,q_X) and (p'_X,q'_X), where $X = L$ for left child and $X = R$ for right child, be the starting clouds of the left and right children of u and $\psi(u)$ respectively. Let c_v be the cost of v, and let $\overline{c_v}$ be the cost of v plus the cost of all the descendants of v. Finally, let c'_v be the cost of $\psi(v)$. An abbreviated version of Theorem 17 (see the full version for extended statement) relates the cost of v with $\psi(v)$ in terms of the starting and ending clouds by defining ψ inductively from the root of $\mathbb{T}_{OPT}$ down. The theorem uses Lemma 16 repeatedly to construct the nodes of $\mathbb{T}$.

▶ **Theorem 17.** *For any approximation algorithm $\mathcal{A} \in \mathcal{F}$ with precision functions α, β, there exists a PET $\mathbb{T} \in \mathcal{D}_{\mathcal{A}}$ and a PET mapping $\psi : V(\mathbb{T}_{OPT}) \to V(\mathbb{T})$ such that $\mathbb{T}_{OPT}$ can be partitioned into disjoint sets $U^1_{NL} \cup U^2_{NL} \cup U^1_L \cup U^2_L \cup X$ with the following properties. For $v \in \mathbb{T}_{OPT}$, if $v \in U^1_{NL} \cup U^2_{NL}$ then v satisfies (Non-leaf), and if $v \in U^1_L \cup U^2_L$ then v satisfies (Leaf):*

$$c'_v \le c_v + \big(|p'_v - p_v| + |q'_v - q_v|\big) - \big(|r'_v - r_v| + |s'_v - s_v|\big) + 2\alpha(r_v, s_v) \qquad \text{(Non-leaf)}$$

$$c'_v \le \overline{c_v} + |p'_v - p_v| + |q'_v - q_v| + \beta(r,s) \qquad \text{(Leaf)}$$

$$\textit{Furthermore } (|p'_L - p_L| + |q'_L - q_L|) + (|p'_R - p_R| + |q'_R - q_R|) \le |r'-r| + |s'-s| + 2\beta(r,s) \quad (*)$$

Let $\mathbb{T}'_{OPT} \subset \mathbb{T}_{OPT}$ be the subgraph of nodes v in the tree for which either v is the only node mapped to $\psi(v) \in \mathbb{T}$, or v is the node closest to the root that is mapped to $\psi(v)$. In the previous theorem, the set X corresponds to the nodes v for which $\psi(v) = \psi(u)$ such that u is an ancestor of v in $\mathbb{T}_{OPT}$. So $\mathbb{T}'_{OPT} = \mathbb{T}_{OPT} \setminus X$. The final theorem is the result of summing over the bounds from Theorem 17 for all $v_j \in \mathbb{T}'_{OPT}$, applying the appropriate bound depending on the set v_j belongs to.

▶ **Theorem 18.** *For any $\mathcal{A} \in \mathcal{F}$ with precision functions α, β, let $\mathbb{T}_{OPT}$ be the PET of any optimal algorithm. Label the nodes of $\mathbb{T}'_{OPT} \subset \mathbb{T}_{OPT}$ by $v_1 \dots v_K$. For $1 \leq i \leq K$, let $(p_i, q_i), (r_i, s_i)$ be the starting and ending clouds of the path v_i in $\mathcal{T}$, then*

$$|OPT| \leq |\mathcal{A}| \leq |OPT| + \sum_{v_j \in \mathbb{T}'_{OPT}} \Big(2\alpha(r_j, s_j) + 3\beta(r_j, s_j)\Big).$$

As an illustration of Theorem 18, consider the γ-uniform grid approximation of Theorem 4. In this case, we have the upper bound $\alpha(r_j, s_j) = \beta(r_j, s_j) = 2\gamma$ for all $v_j \in \mathbb{T}_{OPT}$. Since there are k^* total vertices in $\mathbb{T}_{OPT}$, we get $|OPT| \leq |\mathcal{A}| \leq |OPT| + 10\gamma k^*$. To analyze the running time, note that we only compute non-linear production edges for $(n/\gamma)^2$ clouds, and in each cloud that we compute non-liner edges for we consider at most n/γ break-points. Thus the total runtime is $O(|P|(\frac{n}{\gamma})^3)$ to compute non-linear edges, and $O(|P|n^2)$ to a shortest path algorithm on $\mathcal{T}^{\mathcal{A}}$, for a total runtime of $O(|P|(n^2 + (\frac{n}{\gamma})^3))$.

Our second illustration of Theorem 18 is the γ-non-uniform grid approximation of Theorem 5. Here we obtain a $O(k\gamma)$ additive approximation in time $O(|P|(n^2 + \frac{n^3}{\gamma^2}))$. A detailed analysis can be found in the full version.

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *FOCS 2015*, 2015.
2. Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1974.
3. Alfred V. Aho and Thomas G. Peterson. A minimum distance error-correcting parser for context-free languages. *SIAM J. Comput.*, 1(4), 1972.
4. Rolf Backofen, Dekel Tsur, Shay Zakov, and Michal Ziv-Ukelson. Sparse RNA Folding: Time and Space Efficient Algorithms. In *Annual Symposium on Combinatorial Pattern Matching*, pages 249–262. Springer, 2009.
5. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *STOC 2015*, 2015.
6. Arturs Backurs and Krzysztof Onak. Fast algorithms for parsing sequences of parentheses with few errors. In *PODS*, 2016.
7. Ulrike Brandt and Ghislain Delepine. Weight-reducing grammars and ultralinear languages. *RAIRO-Theoretical Informatics and Applications*, 38(1):19–25, 2004.
8. Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia V. Williams. Truly subcubic algorithms for language edit distance and RNA folding via fast bounded-difference min-plus product. In *FOCS 2016*, 2016.
9. Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *FOCS 2015*, 2015.
10. J. A. Brzozowski. Regular-like expressions for some irregular languages. In *IEEE Annual Symposium on Switching and Automata Theory*, 1968.
11. Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

12 Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13, 1970.

13 Sheila A. Greibach. The unsolvability of the recognition of linear context-free languages. *Journal of the ACM (JACM)*, 13(4):582–587, 1966.

14 Steven Grijzenhout and Maarten Marx. The quality of the XML web. *Web Semant.*, 19, 2013.

15 J. J. Gutell, R. R.and Cannone, Z. Shang, Y. Du, and M. J. Serra. A story: unpaired adenosine bases in ribosomal RNAs. *Journal of Mol Biology*, 2010.

16 John E. Hopcroft and Jeffrey D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley Longman Publishing Co., Inc., 1969.

17 O. H. Ibarra and T. Jiang. On one-way cellular arrays,. *SIAM J. Comput.*, 16, 1987.

18 Russell Impagliazzo and Ramamohan Paturi. Complexity of k-SAT. In *CCC 1999*, pages 237–240, 1999.

19 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *FOCS 1998*, pages 653–662, 1998.

20 Mark Johnson. PCFGs, Topic Models, Adaptor Grammars and Learning Topical Collocations and the Structure of Proper Names. In *ACL 2010*, pages 1148–1157, 2010.

21 Ik-Soon Kim and Kwang-Moo Choe. Error repair with validation in LR-based parsing. *ACM Trans. Program. Lang. Syst.*, 23(4), July 2001.

22 Donald E Knuth. Semantics of context-free languages. *Mathematical systems theory*, 2(2):127–145, 1968.

23 Flip Korn, Barna Saha, Divesh Srivastava, and Shanshan Ying. On repairing structural problems in semi-structured data. In *VLDB 2013*, 2013.

24 Martin Kutriba and Andreas Malcher. Finite turns and the regular closure of linear context-free languages. *Discrete Applied Mathematics*, 155(5), October 2007.

25 Lillian Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49, 2002.

26 Andreas Malcher and Giovanni Pighizzini. Descriptional complexity of bounded context-free languages. *Information and Computation*, 227, June 2013.

27 Christopher D. Manning. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

28 Darnell Moore and Irfan Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *NCAI 2002*, pages 770–776, 2002.

29 E. Moriya and T. Tada. On the space complexity of turn bounded pushdown automata. *Internat. J. Comput*, 80:295—-304., 2003.

30 Gene Myers. Approximately matching context-free languages. *Information Processing Letters*, 54, 1995.

31 Geoffrey K. Pullum and Gerald Gazdar. Natural languages and context-free languages. *Linguistics and Philosophy*, 4(4), 1982.

32 Sanguthevar Rajasekaran and Marius Nicolae. An error correcting parser for context free grammars that takes less than cubic time. *Manuscript*, 2014.

33 Andrea Rosani, Nicola Conci, and Francesco G. De Natale. Human behavior recognition using a context-free grammar. *Journal of Electronic Imaging*, 23(3), 2014.

34 Barna Saha. The Dyck language edit distance problem in near-linear time. In *FOCS 2014*, pages 611–620, 2014.

35 Barna Saha. Language edit distance and maximum likelihood parsing of stochastic grammars: Faster algorithms and connection to fundamental graph problems. In *FOCS 2015*, pages 118–135, 2015.

36 Jose M Sempere and Pedro Garcia. A characterization of even linear languages and its application to the learning problem. In *International Colloquium on Grammatical Inference*, pages 38–44. Springer, 1994.

37 Jeffrey Ullman and John Hopcroft. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

38 Leslie G. Valiant. General context-free recognition in less than cubic time. *Journal of computer and system sciences*, 10(2), 1975.

39 Balaji Venkatachalam, Dan Gusfield, and Yelena Frid. Faster Algorithms for RNA-Folding Using the four-Russians Method. In *WABI 2013*, 2013.

40 Robert A. Wagner. Order-n correction for regular languages. *Communications of the ACM*, 17(5), 1974.

41 Ye-Yi Wang, Milind Mahajan, and Xuedong Huang. A unified context-free grammar and n-gram model for spoken language processing. In *ICASP 2000*, pages 1639–1642, 2000.

42 Glynn Winskel. The formal semantics of programming languages: an introduction, 1993.

43 D. A. Workman. Turn-bounded grammars and their relation to ultralinear languages. *Inform. and Control*, 32:188–200, 1976.

44 Shay Zakov, Dekel Tsur, and Michal Ziv-Ukelson. Reducing the worst case running times of a family of RNA and CFG problems, using Valiant's approach. In *WABI 2010*, 2010.

Conditional Lower Bounds for All-Pairs Max-Flow*

Robert Krauthgamer[1] and Ohad Trabelsi[2]

1 **Weizmann Institute of Science, Rehovot, Israel**
robert.krauthgamer@weizmann.ac.il

2 **Weizmann Institute of Science, Rehovot, Israel**
ohad.trabelsi@weizmann.ac.il

Abstract

We provide evidence that computing the maximum flow value between every pair of nodes in a directed graph on n nodes, m edges, and capacities in the range $[1..n]$, which we call the All-Pairs Max-Flow problem, cannot be solved in time that is faster significantly (i.e., by a polynomial factor) than $O(n^2m)$. Since a single maximum st-flow in such graphs can be solved in time $\tilde{O}(m\sqrt{n})$ [Lee and Sidford, FOCS 2014], we conclude that the all-pairs version might require time equivalent to $\tilde{\Omega}(n^{3/2})$ computations of maximum st-flow, which strongly separates the directed case from the undirected one. Moreover, if maximum st-flow can be solved in time $\tilde{O}(m)$, then the runtime of $\tilde{\Omega}(n^2)$ computations is needed. This is in contrast to a conjecture of Lacki, Nussbaum, Sankowski, and Wulf-Nilsen [FOCS 2012] that All-Pairs Max-Flow in general graphs can be solved faster than the time of $O(n^2)$ computations of maximum st-flow.

Specifically, we show that in sparse graphs $G = (V, E, w)$, if one can compute the maximum st-flow from every s in an input set of sources $S \subseteq V$ to every t in an input set of sinks $T \subseteq V$ in time $O((|S||T|m)^{1-\varepsilon})$, for some $|S|$, $|T|$, and a constant $\varepsilon > 0$, then MAX-CNF-SAT (maximum satisfiability of conjunctive normal form formulas) with n' variables and m' clauses can be solved in time ${m'}^{O(1)}2^{(1-\delta)n'}$ for a constant $\delta(\varepsilon) > 0$, a problem for which not even $2^{n'}/\mathsf{poly}(n')$ algorithms are known. Such runtime for MAX-CNF-SAT would in particular refute the Strong Exponential Time Hypothesis (SETH). Hence, we improve the lower bound of Abboud, Vassilevska-Williams, and Yu [STOC 2015], who showed that for every fixed $\varepsilon > 0$ and $|S| = |T| = O(\sqrt{n})$, if the above problem can be solved in time $O(n^{3/2-\varepsilon})$, then some incomparable (and intuitively weaker) conjecture is false. Furthermore, a larger lower bound than ours implies strictly super-linear time for maximum st-flow problem, which would be an amazing breakthrough.

In addition, we show that All-Pairs Max-Flow in *uncapacitated* networks with every edge-density $m = m(n)$, cannot be computed in time significantly faster than $O(mn)$, even for acyclic networks. The gap to the fastest known algorithm by Cheung, Lau, and Leung [FOCS 2011] is a factor of $O(m^{\omega-1}/n)$, and for acyclic networks it is $O(n^{\omega-1})$, where ω is the matrix multiplication exponent.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Conditional lower bounds, Hardness in P, All-Pairs Maximum Flow, Strong Exponential Time Hypothesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.20

* This work was partially supported by the Israel Science Foundation grant #897/13 and by a Minerva Foundation grant.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 20; pp. 20:1–20:13

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

1 Introduction

The maximum flow problem is one of the most fundamental problems in combinatorial optimization. This classic problem and its variations such as minimum-cost flow, integral flow, and minimum-cost circulation, were studied extensively over the past decades, and have become key algorithmic tools with numerous applications, in theory and in practice. Moreover, techniques developed for flow problems were generalized or adapted to other problems, see for example [6, 3, 5]. The maximum st-flow problem, which we shall denote Max-Flow, asks to ship the maximum amount of flow from a source node s to a sink node t in a directed edge-capacitated graph $G = (V, E, w)$, where throughout, we denote $n = |V|$ and $m = |E|$, and assume integer capacities bounded by U. After this problem was introduced in 1954 by Harris and Ross (see [22] for a historical account), Ford and Fulkerson [12] devised the first algorithm for Max-Flow, which runs in time $O((n+m)F)$, where F is the maximum value of a feasible flow. Ever since, a long line of generalizations and improvements was studied, and the current fastest algorithm for Max-Flow with arbitrary capacities is by Lee and Sidford [20], which takes $O(m\sqrt{n}\log U)$ time. For the case of small capacities and sufficiently sparse graphs, the fastest algorithm, due to Mądry [21], has a running time $\tilde{O}(m^{10/7}U^{1/7})$. Here and throughout, $\tilde{O}(f)$ denotes $O(f \log^c f)$ for unspecified constant $c > 0$.

A very natural problem is to compute the maximum st-flow for multiple source-sink pairs in the same graph G. The seminal work of Gomory and Hu [14] shows that in undirected graphs, Max-Flow for all $\binom{n}{2}$ source-sink pairs requires at most $n-1$ executions of Max-Flow (see also [15], where the $n-1$ computations are all on the input graph), and a lot of research aimed to extend this result to directed graphs, with several partial successes, see details in Section 1.1. However, it is still not known how to solve Max-Flow for multiple source-sink pairs faster than solving it separately for each pair, even in special cases like a single source and all possible sinks. We shall consider the following problems involving multiple source-sink pairs, where the goal is always to report the value of each flow (and not an actual flow attaining it).

▶ **Definition 1.1.** (Single-Source Max-Flow) Given a directed edge-capacitated graph $G = (V, E, w)$ and a source node $s \in V$, output, for every $t \in V$, the maximum flow that can be shipped in G from s to t.

▶ **Definition 1.2.** (All-Pairs Max-Flow) Given a directed edge-capacitated graph $G = (V, E, w)$, output, for every pair of nodes $u, v \in V$, the maximum flow that can be shipped in G from u to v.

▶ **Definition 1.3.** (ST-Max-Flow) Given a directed edge-capacitated graph $G = (V, E, w)$ and two subsets of nodes $S, T \subseteq V$, output, for every pair of nodes $s \in S$ and $t \in T$, the maximum flow that can be shipped in G from s to t.

▶ **Definition 1.4.** (Global Max-Flow) Given a directed edge capacitated graph $G = (V, E, w)$, output the maximum among all pairs $u, v \in V$, of the maximum flow value that can be shipped in G from u to v.

▶ **Definition 1.5.** (Maximum Local Edge Connectivity) Given a directed graph $G = (V, E)$, output the maximum among all pairs $u, v \in V$, of the maximum number of edge-disjoint uv-paths in G.

Note that in a graph with all edge capacities equal to 1, the problem of finding the maximum local edge connectivity is equivalent to finding the global maximum flow.

Table 1 Known algorithms for multiple-pairs Max-Flow. In this table, $T(n,m)$ is the fastest time to compute maximum st-flow in an undirected graph, ω is the matrix multiplication exponent, and $\gamma = \gamma(G)$ is a topological property of the input network that varies between 1 and $\Theta(n)$. In planar graphs, γ is the minimum number of faces required to cover all the nodes (i.e., every node is adjacent to at least one such face) over all possible planar embeddings [13].

Directed	Class	Problem	Runtime	Reference
No	General	All-Pairs (G-H Tree)	$(n-1)T(n,m)$	[14]
No	Uncapacitated Networks	All-Pairs (G-H Tree)	$\tilde{O}(mn)$	[7]
No	Genus bounded by g	All-Pairs (G-H Tree)	$2^{O(g^2)} n \log^3 n$	[8]
Yes	Sparse	All-Pairs	$O(n^2 + \gamma^4 \log \gamma)$	[4]
Yes	Constant Treewidth	All-Pairs	$O(n^2)$	[4]
Yes	Uncapacitated	All-Pairs	$O(m^\omega)$	[11]
Yes	Uncapacitated DAG	Single-Source	$O(n^{\omega-1}m)$	[11]
Yes	Planar	Single-Source	$O(n \log^3 n)$	[19]

1.1 Prior Work

We start with undirected graphs, where the All-Pairs Max-Flow values can be represented in a very succint manner, called nowdays *a Gomory-Hu* tree [14]. In addition to being very succint, it allows the flow values and the corresponding cuts (vertex partitions) to be quickly retrieved. See Table 1 for a list of previous algorithms for multiple pairs maximum st-flow, see Table 1. For directed graphs, no current algorithm computes the maximum flow between any $k = \omega(1)$ given pairs of nodes faster than the time of $O(k)$ separate Max-Flow computations. However, some results are known in special settings. It is possible to compute Max-Flow for $O(n)$ pairs in the time it takes for a single Max-Flow computation [16] and this result is used to find a global minimum cut. However, these pairs cannot be specified in the input.

For directed planar graphs, there is an $O(n \log^3 n)$ time algorithm for the Single-Source Max-Flow problem [19], which immediately yields an $O(n^2 \log^3 n)$ time algorithm for the All-Pairs version, that is much faster than the time of $O(n^2)$ computations of planar Max-Flow, a problem that can be solved in time $O(n \log n)$ [9]. Based on these results, it was conjectured in [19] that also in general graphs, All-Pairs Max-Flow can be solved faster than the time required for computing $O(n^2)$ separate maximum st-flows.

Several hardness results are known for multiple-pairs variants of Max-Flow [2]. For ST-Max-Flow in sparse graphs ($m = O(n)$) and $|S| = |T| = O(\sqrt{n})$, there is an $n^{3/2-o(1)}$ lower bound assuming at least one of the Strong Exponential Time Hypothesis (SETH), 3SUM, and All-Pairs Shortest-Paths (APSP) conjectures is correct (for a comprehensive survey on them, see [23]). In addition, they show that Single-Source Max-Flow on sparse graphs requires $n^{2-o(1)}$ time, unless MAX-CNF-SAT can be solved in time $2^{(1-\delta)n}\mathsf{poly}(m)$ for some fixed $\delta > 0$, and in particular SETH is false.

We will mostly rely on SETH, a conjecture introduced by [17], and on some weaker assumption related to its maximization version, MAX-CNF-SAT. In more detail, SETH states that for every fixed $\varepsilon > 0$ there is an integer $k \geq 3$ such that k-SAT on n variables and m clauses cannot be solved in time $2^{(1-\varepsilon)n}\mathsf{poly}(m)$, where $\mathsf{poly}(m)$ refers to $O(m^c)$ for unspecified constant c. By the sparsification lemma [18], in order to refute SETH it can be assumed that the number of clauses is $O(n)$. The MAX-CNF-SAT problem asks for the maximum number of clauses that can be satisfied in an input CNF formula. Most

of our conditional lower bounds are based on the assumption that for every fixed $\delta > 0$, MAX-CNF-SAT cannot be solved in time $2^{(1-\delta)n}\mathsf{poly}(m)$, where currently even $2^n/\mathsf{poly}(n)$ algorithms are not known for this problem [2]. Note that this is a weaker assumption than SETH, since a faster algorithm for MAX-CNF-SAT would imply a faster algorithm for CNF-SAT and refute SETH. Different assumptions regarding the hardness of CNF-SAT have been the basis for many lower bounds, including for the runtime of solving NP-hard problems exactly, parametrized complexity, and problems in P. See the Introduction in [1] and the references therein.

1.2 Our Contribution

We present conditional runtime lower bounds for both uncapacitated and capacitated networks. The proofs appear in sections 2 and 3, respectively, where the order reflects increasing level of complication. All our lower bounds hold even when the input G is a DAG and has a constant diameter, and in the case of general capacities, they can be easily modified to apply also for graphs with constant maximum degree. In addition, for integer $k \geq 1$ we use the notation $[k]$ to denote the range $\{1, ..., k\}$.

Capacitated Networks

Our main result is that for every set sizes $|S|$ and $|T|$, the ST-Max-Flow cannot be solved significantly faster than $O(|S||T|m)$ (i.e., polynomially smaller runtime), unless a breakthrough in MAX-CNF-SAT is achieved, and consequently in SETH.

▶ **Theorem 1.6.** *If for some fixed $\varepsilon > 0$ and some (possibly functions of n) set sizes $|S|$ and $|T|$, ST-Max-Flow can be solved in graphs with n nodes, $m = O(n)$ edges and capacities in $[n]$ in time $O((|S||T|m)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

This result improves the aforementioned $n^{3/2-o(1)}$ lower bound of [2], as for their setting of $|S| = |T| = O(\sqrt{n})$ our lower bound is $n^{2-o(1)}$, although their lower bound is based on an incomparable (and intuitively weaker) conjecture, that at least one of the SETH, 3SUM, and APSP conjectures is correct. In fact, if there was a reduction from SETH that implied a larger runtime lower bound for ST-Max-Flow, then the (single-pair) Max-Flow problem would require a strictly super-linear time under it, but such a reduction is not possible unless the non-deterministic version of SETH (abbreviated NSETH) is false [10]. And anyway, such a lower bound for Max-Flow would be an amazing breakthrough.

The next theorem is an immediate corollary of Theorem 1.6, by assigning $|S|, |T| = \Theta(n)$.

▶ **Theorem 1.7.** *If for some fixed $\varepsilon > 0$, All-Pairs Max-Flow in graphs with n nodes, $m = O(n)$ edges, and capacities in $[n]$ can be solved in time $O((n^2m)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

This conditional lower bound (see Figure 1) shows that All-Pairs Max-Flow requires time that is equivalent to $\Omega(n^{3/2})$ computations of Max-Flow, which strongly separates the directed case from the undirected one (where a Gomory-Hu tree can be constructed in the time of $n-1$ computations). If Max-Flow takes $\tilde{O}(m)$ time, which is currently open but plausible, then the running time of $\tilde{\Omega}(n^2)$ computations of Max-Flow is needed. This is in contrast to the aforementioned conjecture of Lacki, Nussbaum, Sankowski, and Wulf-Nilsen [19] that All-Pairs Max-Flow in general graphs can be solved faster than the time of $O(n^2)$ computations of maximum st-flow.

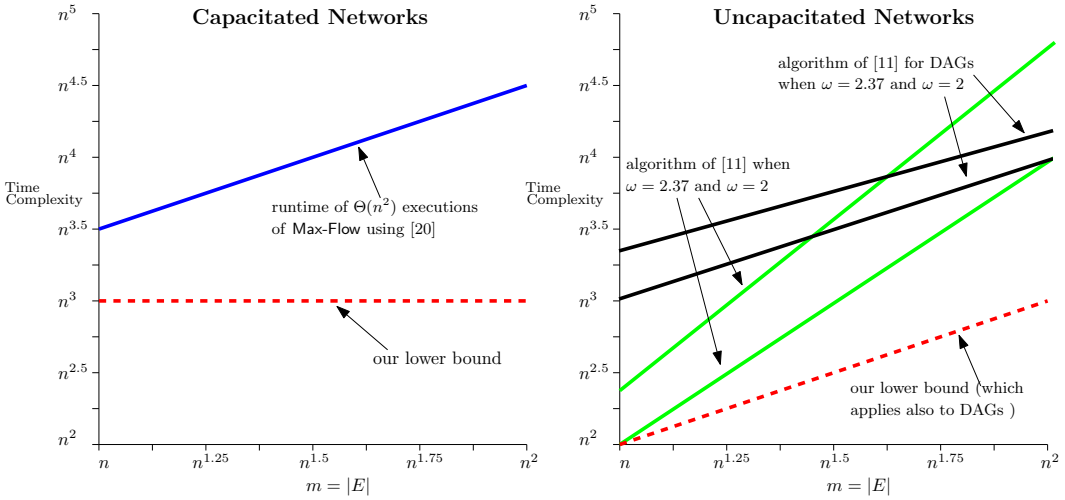


Figure 1 State of the art bounds for All-Pairs Max-Flow in directed networks. Conditional lower bounds are depicted in dashed lines, and known algorithms in solid lines.

Uncapacitated Networks

For the case of uncapacitated networks, we show that for every $m = m(n)$, All-Pairs Max-Flow cannot be solved significantly faster than $O(mn)$. Here we introduce a new technique to design reductions from SETH to graphs with varying edge densities, rather than the usual reductions that only deal with sparse graphs. Our technique is based on partitioning the variables set of CNF-SAT to different sizes.

▶ **Theorem 1.8.** *If for some fixed $\varepsilon > 0$ and some $m = m(n) \in [n, n^2]$, All-Pairs Max-Flow in uncapacitated graphs with n nodes and m edges can be solved in time $O((nm)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

Hence, a certain additional improvement to the $O(m^\omega)$ time algorithm of [11] (and similarly to the $O(n^\omega m)$ time for DAGs, where our lower bounds apply too) is not likely. We now present conditional lower bounds for ST-Max-Flow, which are functions of $|S|$ and $|T|$.

▶ **Theorem 1.9.** *If for some fixed $\varepsilon > 0$ and some (possibly functions of n) set sizes $|S|$ and $|T|$, ST-Max-Flow on uncapacitated graphs with n nodes and $O((|S| + |T|)n)$ edges can be solved in time $O((|S||T|n)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

Finally, we present a conditional lower bound for computing the Maximum Local Edge Connectivity of a sparse graph, which is the same as Global Max-Flow if all the capacities are 1, which is indeed the case in our reduction. In the Orthogonal Vectors problem the input is two sets U and V, each of n vectors from $\{0,1\}^d$, and the goal is to determine whether there are $u \in U$ and $v \in V$ such that $\sum_{i=1}^{d} u_i \cdot v_i = 0$. An equivalent version of the problem has $U = V$. For $d = \omega(\log n)$, Williams [24] proved that SETH implies the non-existence of a truly subquadratic (in n) algorithm for the problem. The next result, proved in Section 4, was obtained together with Bundit Laekhanukit and Rajesh Chitnis, and we thank them for their permission to include it here.

▶ **Theorem 1.10.** *If for some fixed $\varepsilon > 0$, the Maximum Local Edge Connectivity in graphs with n nodes and $\tilde{O}(n)$ edges can be found in time $O(n^{2-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, the Orthogonal Vectors problem on n' vectors and every $d = \mathsf{polylog}(n')$ can be solved in time $O(n'^{2-\delta})$, and in particular SETH is false.*

2 Reduction to Multiple-Pairs Max-Flow with Unit Capacity

In this section we prove Theorems 1.8 and 1.9. We start with a general lemma which is the heart of the proofs.

▶ **Lemma 2.1.** *Let $a \in [0,1]$ and $b \in [0, 1-a]$. Then MAX-CNF-SAT on n variables and m clauses can be reduced to $O(m)$ instances of ST-Max-Flow with $|S| = 2^{an}$ and $|T| = 2^{bn}$ in graphs with $\Theta(2^{an} + 2^{(1-a-b)n}m + 2^{bn})$ nodes, $\Theta((2^{an} + 2^{bn}) \cdot 2^{(1-a-b)n}m)$ edges, and capacities in $\{0,1\}$.*

Proof. Given a CNF-formula F on n variables and m clauses $\{C_i\}_{i\in[m]}$ as input for MAX-CNF-SAT, $a \in [0,1]$, and $b \in [0, 1-a]$, we split the variables into three sets U_1, U_2, and U_3, where U_1 is of size an, U_2 is of size $(1-a-b)n$, and U_3 is of size bn, and enumerate all their 2^{an}, $2^{(1-a-b)n}$, and 2^{bn} partial assignments (with respect to F), respectively, when the objective is to find a triple α, β, γ of assignments to U_1, U_2, and U_3 respectively, that satisfies the maximal number of clauses. We will have an instance G_p of ST-Max-Flow for each value $p \in [m]$, in which by one call to ST-Max-Flow we check if there exists a triple α, β, and γ that satisfies at least p clauses, as follows.

We construct a graph G_p for every $p \in [m]$ on N nodes $V_1 \cup V_2 \cup V_3$, where V_1 contains a node α for every assignment α to U_1, V_2 contains $2m + 1 + (p-1) = 2m + p$ nodes for every assignment β to U_2, that are β_i^l and β_i^r for every $i \in [m]$, β', and the set $\{\beta'^i\}_{i\in[p-1]}$, and V_3 contains a node γ for every assignment γ to U_3. We use the notation α for nodes in V_1 and for assignments to U_1, β for assignments to U_2, and γ for nodes in V_3 and assignments to U_3. However, it will be clear from the context. Now, we have to describe the edges in the network. In order to simplify the reduction, we partition the edges into blue and red colors, as follows.

For every α, β, and $i \in [m]$, we add a blue edge from α to β_i^l if both of α and β do not satisfy the clause C_i (do not set any of the literals to true), and otherwise we add a red edge from α to β_i^r. We further add, for every β, γ, and $i \in [m]$, a blue edge from β_i^l to γ if γ does not satisfy C_i. For every β, γ, and $j \in [p-1]$, we add a red edge from every β'^j to every γ. For every β and $i \in [m]$, we add a red edge from β_i^l to β_i^r and from β_i^r to β', and finally for every β and $j \in [p-1]$, we add a red edge from β' to β'^j, where all edges are of capacity 1.

The graph we built has $2^{an} + 2 \cdot 2^{(1-a-b)n}m + 2^{(1-a-b)n} + 2^{(1-a-b)n}(p-1) + 2^{bn} = \Theta(2^{an} + 2^{(1-a-b)n}m + 2^{bn})$ nodes, $2^{an} \cdot 2^{(1-a-b)n}m + 2^{bn} \cdot 2^{(1-a-b)n}m + 2 \cdot 2^{(1-a-b)n}m + (p-1)2^{(1-a-b)n} + 2^{bn} \cdot (p-1)2^{(1-a-b)n} = \Theta((2^{an} + 2^{bn}) \cdot 2^{(1-a-b)n}m)$ edges, with capacities in $\{0,1\}$ (see Figure 2), and its construction time is asymptotically the same as the time it takes to construct its edges set.

For every α, β, and γ, we denote by $G_p^{\alpha,\beta,\gamma}$ the graph induced from G_p on the nodes

$$(\alpha, \beta', \gamma) \cup \left(\bigcup_{\substack{y\in\{l,r\} \\ i\in[m]}} \beta_i^y \right) \cup \left(\bigcup_{j\in[p-1]} \beta'^j \right).$$

We claim that for every α and γ, the maximum flow from α to γ can be bounded by the sum, over all β, of the maximum flow between them in $G_p^{\alpha,\beta,\gamma}$. This claim follow easily

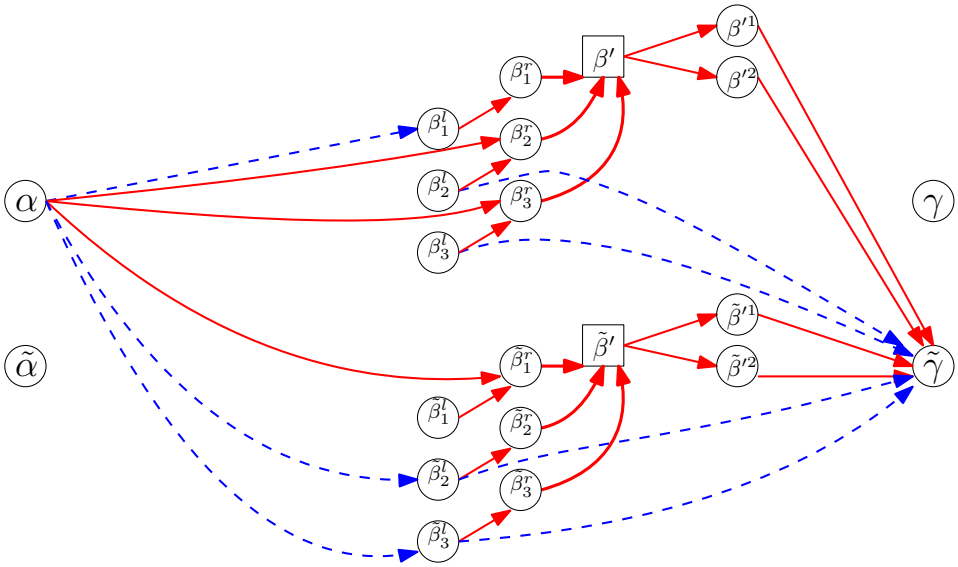


Figure 2 An illustration of part of the reduction. Here, U_1, U_2, and U_3 have 2 assignments each, α and $\tilde{\alpha}$ to U_1, β and $\tilde{\beta}$ to U_2, and γ and $\tilde{\gamma}$ to U_3. Blue edges are dashed. For simplicity, only the edges of $G_3^{\alpha,\beta,\tilde{\gamma}} \cup G_3^{\alpha,\tilde{\beta},\tilde{\gamma}}$ are presented. In this illustration, α does not satisfy anything, β satisfies C_2 and C_3, $\tilde{\beta}$ satisfies C_1, and $\tilde{\gamma}$ satisfies C_1. Note that the assignment comprised of α, β, and $\tilde{\gamma}$ satisfies all the clauses, and indeed the maximum flow from α to γ is $2 \cdot 3 - 1 = 5$.

because the intersection $G_p^{\alpha,\beta_1,\gamma} \cap G_p^{\alpha,\beta_2,\gamma}$ for $\beta_1 \neq \beta_2$ is exactly the source and the sink $\{\alpha, \gamma\}$, no edge passes between these two graphs, and $\left(\bigcup_\beta G_i^{\alpha,\beta,\gamma}\right)$ consists of all nodes that are both reachable from α and γ is reachable from them.

We now prove that if there is an assignment to F that satisfies at least p clauses then the graph G_p we built has a triple α, β, γ with maximum flow from α to γ in $G_p^{\alpha,\beta,\gamma}$ at most $m-1$. Since for every $\tilde{\beta}$, m is the number of outgoing edges from α in $G_p^{\alpha,\tilde{\beta},\gamma}$, m is also an upper bound for the maximum flow from α to γ in it, and hence in G_p it is at most $2^{(1-a-b)n}m - 1$. Otherwise, we will show that every triple α, β, γ has a maximum flow from α to γ in $G_p^{\alpha,\beta,\gamma}$ of size at least m, and so in G_p it is at least $2^{(1-a-b)n}m$. Hence, by simply picking the maximal $j \in [m]$ such that the maximum flow in G_j of some pair α, γ is at most $2^{(1-a-b)n}m - 1$, and then iterating over all assignments β to U_2 with α and γ fixed as the assignments to U_1 and U_3, we can also find the required triple α, β, γ.

For the first direction, assume that F has an assignment that satisfies at least p clauses, and denote such assignment by Φ. Let α_Φ, β_Φ, and γ_Φ be the assignments to U_1, U_2, and U_3, respectively, that are induced from Φ. Since a blue path from α_Φ through $\beta_{\Phi_i}^l$ for some $i \in [m]$ to γ_Φ corresponds to α_Φ, β_Φ, and γ_Φ all do not satisfy C_i, in $G_p^{\alpha_\Phi,\beta_\Phi,\gamma_\Phi}$ there are at most $m - p$ (internally) disjoint blue paths from α to γ. As the only way to ship flow in $G_p^{\alpha_\Phi,\beta_\Phi,\gamma_\Phi}$ that is not through a blue path is through the node β'_Φ, and the total number of edges going out of this node is $p-1$, we conclude that the total maximum flow in $G_p^{\alpha_\Phi,\beta_\Phi,\gamma_\Phi}$ from α_Φ to β_Φ is bounded by $m - p + (p-1) = m - 1$. Since for every β, the maximum amount of flow that can be shipped in $G_p^{\alpha_\Phi,\beta,\gamma_\Phi}$ from α_Φ to γ_Φ is at most m, summing over all β we get that the total flow in G_p from α_Φ to γ_Φ is bounded by $(2^{(1-a-b)n} - 1)m + (m-1) \leq 2^{(1-a-b)n}m - 1$, as required.

For the second direction, assume that every assignment to F satisfies at most $p-1$ clauses. In order to show that the maximum flow from every α to every γ is at least $2^{(1-a-b)n}m$,

we first fix α, β, and γ. Then, by passing flow in two phases we show that m units of flow can be passed in $G_p^{\alpha,\beta,\gamma}$ from α to γ. As this argument applies for every β, we can add up the respective flows without violating capacities, concluding the proof. By the assumption, there exist $m-(p-1)=m-p+1$ i's, such that α, β, and γ do not satisfy C_i, and we denote a set with this amount of such i's by I_β. Each of these i's induces a blue path $(\alpha \to \beta_i^l \to \gamma)$ from α to γ in $G_p^{\alpha,\beta,\gamma}$, and so we ship a unit of flow through every one of them according to I_β, in what we call the first phase. In the second phase, we ship additional $m-(m-p+1)=p-1$ units in the following way. Let $A_1 := \{i \in [m] \setminus I_\beta : \alpha \nvDash C_i \wedge \beta \nvDash C_i\}$, and $A_2 := ([m] \setminus I_\beta) \setminus A_1 = \{i \in [m] \setminus I_\beta : \alpha \vDash C_i \vee \beta \vDash C_i\}$, where $\alpha \vDash C_i$ denotes that the assignment α satisfies C_i (as defined earlier), and $\alpha \nvDash C_i$ denotes that it does not satisfy C_i. Let $f : A_1 \cup A_2 \to [m-|I_\beta|]$ be a bijective function such that the range of A_1 is $[|A_1|]$ and the range of A_2 is $[m-|I_\beta|] \setminus [|A_1|]$. Clearly, there exists such bijection and it is easy to find one. For every $i \in A_1$ we ship flow through the path $(\alpha \to \beta_i^l \to \beta_i^r \to \beta' \to \beta'^j \to \gamma)$, and for every $i \in A_2$ through the path $(\alpha \to \beta_i^r \to \beta' \to \beta'^j \to \gamma)$, in both cases with $j = f(i)$.

Since we defined the flow in paths, we only need to show that the capacity requirements hold, and we start with blue edges. Indeed, edges of the form (α, β_i^l) are used in the first phase, with flow that is determined uniquely by β and $i \in I_\beta$, and in the second phase uniquely according to β and $i \in [m] \setminus I_\beta$, and so they cannot be used twice. Edges of the form (β_i^l, γ) are only used in the first phase, and their flow is uniquely determined according to β and $i \in I_\beta$, and so are good too. We now proceed to red edges, which were used only in the second phase.

Edges of the forms (α, β_i^r), (β_i^l, β_i^r) and (β_i^r, β') have flow that is uniquely determined by β and $i \in [m] \setminus I_\beta$, and so are not used more than once. Edges of the form (β', β'^j) have flow that is uniquely determined by β and $j = f(i) \in [p-1]$, and since f is a bijection, every j has at most one i such that $f(i) = j$, and so these edges are also used at most once. As a byproduct, and since every edge of the form (β'^j, γ) has only the edge (β', β'^j) as its source for flow, edges of the form (β'^j, γ) are also used at most once. Altogether, we have bounded the total flow in all edges that were used in both phases, and so the capacity requirements follow, which completes the proof of the second direction and of Lemma 2.1. ◀

Proof of Theorem 1.8. We apply Lemma 2.1 in the following way. By setting $a = b \in [1/3, 1/2]$ we get graphs $G = (V, E, w)$ with $|V| = \Theta(2^{an})$ ($|V| = \Theta(2^{an})m$ if $a = 1/3$) and $|E| = 2^{(1-a)n}m$. Hence, $|E| = O(|V|^{1/a-1})$ and we get our desired bound for every $|E|$ between $|V|$ and $|V|^2$ and Theorem 1.8 follows. ◀

Proof of Theorem 1.9. Here we apply Lemma 2.1 a bit differently. By setting $a, b \leq 1/3$ we get graphs $G = (V, E, w)$ with $|V| = \Theta(2^{(1-a-b)n}m)$ and $|E| = \Theta((2^{an} + 2^{bn})2^{(1-a-b)n}m)$. By setting $|S| = |V|^{a/(1-a-b)}$ and $|T| = n^{b/(1-a-b)}$ we get our lower bound for $|E| = O((|S| + |T|)|V|)$ and Theorem 1.9 follows. ◀

3 Reduction to Multiple-Pairs Max-Flow in Capacitated Networks

In this section we prove Theorems 1.6 and 1.7. We proceed to prove our main technical lemma.

▶ **Lemma 3.1.** *MAX-CNF-SAT on n variables and m clauses $\{C_i\}_{i\in[m]}$ can be reduced to $O(m)$ instances of ST-Max-Flow, each with the property that $S \cap T = \emptyset$, and all of them are in graphs with $N = \Theta(2^{n/3}m)$ nodes, $O(2^{n/3}m) = O(N)$ edges, and with capacities in $[N]$.*

Proof. Given a CNF-formula F on n variables and m clauses as input for MAX-CNF-SAT, we split the variables into three sets U_1, U_2, and U_3 of size $n/3$ each and enumerate all $2^{n/3}$ partial assignments (with respect to F) to each of them, when the objective is to find a triple (α, β, γ) of assignments to U_1, U_2, and U_3, that satisfy the maximal number of clauses. We will have an instance G_p of ST-Max-Flow with the mentioned property for each value $p \in [m]$, in which by one call to ST-Max-Flow we check if there exists a triple (α, β, γ) that satisfies at least p clauses, as follows.

We construct the graph G_p on N nodes $V_1 \cup V_2 \cup V_3 \cup A \cup B \cup \{v_B\}$, where V_1 contains a node α for every assignment α to U_1, V_2 contains $3m+1$ nodes for every assignment β to U_2, that are β_i^l, β_i^c, β_i^r, for every $i \in [m]$, and β', V_3 contains a node γ for every assignment γ to U_3, A contains two nodes $C_i^{\models}$ and $C_i^{\not\models}$ for every clause C_i, and B contains a node C_i for every clause C_i. We use the notation α for nodes in V_1 and assignments to U_1, β to assignments to U_2, γ for nodes in V_3 and assignments to U_3, and C_i for nodes in B and clauses. However, it will be clear from the context. Now, we have to describe the edges in the network. In order to simplify the reduction, we partition the edges into red and blue colors, as follows.

For every α and $i \in [m]$ we add a red edge of capacity $2^{n/3}$ from α to $C_i^{\models}$ if $\alpha \models C_i$, and a blue edge of the same capacity from α to $C_i^{\not\models}$ otherwise. We further add, for every β, a red edge of capacity 1 from $C_i^{\models}$ to β_i^c, a blue edge of capacity 1 from $C_i^{\not\models}$ to β_i^l, a blue edge of capacity 1 from β_i^l to β_i^r if $\beta \not\models C_i$, a red edge of capacity 1 from β_i^c to β', and a blue edge of capacity 1 from β_i^r to C_i. For every β we add a red edge of capacity $p-1$ from β' to v_B. For every γ we add a red edge of capacity $2^{n/3}(p-1)$ from v_B to $\gamma \in V_3$, and finally, for every γ and $i \in [m]$ we add a blue edge of capacity $2^{n/3}$ from C_i to γ if $\gamma \not\models C_i$.

The graph we built has $N = 2^{n/3} + 2m + 2^{n/3} \cdot 3m + 2^{n/3} + 1 + m + 2^{n/3} = \Theta(2^{n/3}m)$ nodes, at most $2^{n/3}m + 2^{n/3} \cdot 2m + 2^{n/3} \cdot 2m + 2^{n/3}m + 2^{n/3} + 1 + 2^{n/3}m + 2^{n/3}m = O(2^{n/3}m)$ edges, all of its capacities are in $[N]$, and its construction time is $O(Nm)$ (see Figure 3).

We proceed to prove that if there is an assignment to F that satisfies at least p clauses then the graph G_p we built has a pair α, γ with maximum flow from α to γ at most $2^{n/3}m - 1$, and otherwise, every α, γ has a maximum flow of size at least $2^{n/3}m$. Hence, by simply picking the maximal $j \in [m]$ such that the maximum flow in G_j of some pair α, γ is at most $2^{n/3}m - 1$, and then iterating over all assignments β to U_2 with α and γ fixed as the assignments to U_1 and U_3, we can also find the required triple α, β, γ.

For the first direction, assume that F has an assignment that satisfies at least p clauses, and denote such assignment by Φ. Let α_Φ, β_Φ, and γ_Φ be the assignments to U_1, U_2, and U_3, respectively, that are induced from Φ. We will show that there exists an $(\alpha_\Phi, \gamma_\Phi)$ cut whose capacity is at most $2^{n/3}m - 1$, hence by the Min-Cut Max-Flow theorem, the maximum flow from α_Φ to γ_Φ is bounded by this number, concluding the proof of the first direction. We define the cut in a way that for every $\beta \neq \beta_\Phi$, the cut will have m cut edges that are contributed from nodes related to β, and nodes related to β_Φ will be carefully added to either side of the cut so that they will contribute capacity of only $m-1$ to the cut. To be more precise, we define a suitable cut as follows.

$$S = \{\alpha_\Phi, \beta'_\Phi\} \cup \{C_i^{\models} : \alpha_\Phi \models C_i\} \cup \{C_i^{\not\models} : \alpha_\Phi \not\models C_i\} \cup \{{\beta_\Phi}_i^c : i \in [m]\}$$
$$\cup \{C_i, {\beta_\Phi}_i^l, {\beta_\Phi}_i^r : \gamma_\Phi \models C_i\} \cup \{{\beta_\Phi}_i^l : \gamma_\Phi \not\models C_i \wedge \beta_\Phi \models C_i\}$$

▶ **Claim 3.2.** *The cut $(S, V \setminus S) = (S, T)$ has capacity $2^{n/3}m - 1$.*

Proof of Claim. We will go over all the nodes in S, and count the total capacity leaving to nodes in T for each of them. $\alpha_\Phi \in S$ and all nodes $C_i^{\models}$ and $C_i^{\not\models}$ that are adjacent to it are in S too, hence it does not contribute anything. For every $i \in [m]$, we have two cases for

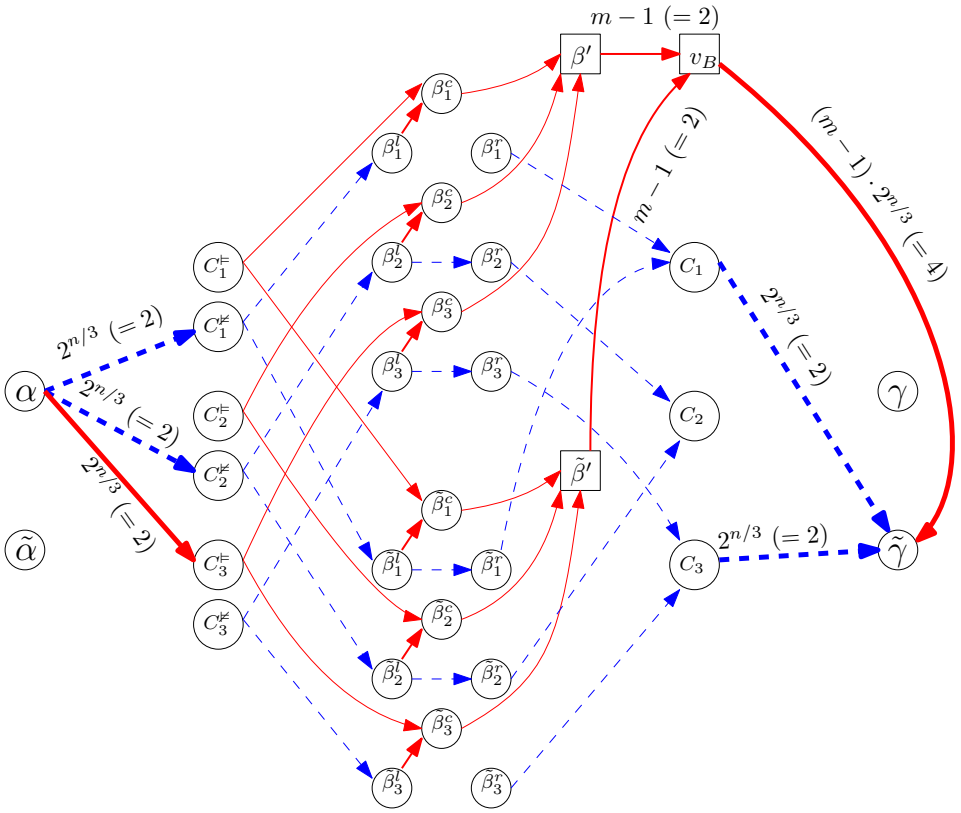


Figure 3 An illustration of part of the reduction, with $p = m$. Here, U_1, U_2, and U_3 have 2 assignments each; α and $\tilde{\alpha}$ to U_1, β and $\tilde{\beta}$ to U_2, γ and $\tilde{\gamma}$ to U_3. Bolder edges correspond to edges of higher capacity (specified wherever they are bigger than 1), and blue edges are dashed. For simplicity, only the edges relevant to α and $\tilde{\gamma}$ are presented. In this illustration, α satisfies C_3, β satisfies C_1, $\tilde{\beta}$ satisfies C_3, and $\tilde{\gamma}$ satisfies C_2. Note that the assignment comprised of α, β, and $\tilde{\gamma}$ satisfies all the clauses, and indeed the maximum flow from α to γ is $2 \cdot 3 - 1 = 5$.

nodes in A. If $\alpha_\Phi \vDash C_i$ then $C_i^{\nvDash} \in T$ and hence $C_i^{\nvDash}$ does not contribute anything. However, $C_i^{\vDash}$ has $2^{n/3}$ outgoing edges, where all except ${\beta_\Phi}_i^c$ are in T. Hence, it contributes $2^{n/3} - 1$ to the cut. Else, if $\alpha_\Phi \nvDash C_i$ then $C_i^{\vDash} \in T$ and hence $C_i^{\vDash}$ does not contribute anything. But $C_i^{\nvDash}$ has $2^{n/3}$ outgoing edges, of which $2^{n/3} - 1$ are cut edges as their targets are in T, and the one incoming to ${\beta_\Phi}_i^l$ is a cut edge if and only if ${\beta_\Phi}_i \nvDash C_i$ and also $\gamma_\Phi \nvDash C_i$ (equivalently, ${\beta_\Phi}_i^l \in T$), and in our current case it means that $\Phi \nvDash C_i$. Hence, for every $i \in [m]$, the nodes in $\{C_i^{\vDash}, C_i^{\nvDash}\}$ contribute $2^{n/3} - 1$ to the cut if $\Phi \vDash C_i$, and $2^{n/3}$ otherwise. Since there are at most $m - p$ clauses that are not satisfied by Φ, summing over all $i \in [m]$ would yield a total of at most $p(2^{n/3} - 1) + (m - p)(2^{n/3}) = 2^{n/3}m - p$ cut edges for vertices with origin in A.

For every $\beta \neq \beta_\Phi$, all nodes in V_2 that are associated with β, v_B, and γ_Φ, are in T and hence will not contribute anything to the cut. However, the node ${\beta_\Phi}'$ is always in S, with v_B its sole target, and hence the edge $({\beta_\Phi}', v_B)$ is in the cut and ${\beta_\Phi}'$ contributes an additional amount of $p - 1$, to a current total of at most $2^{n/3}m - p + (p - 1) = 2^{n/3}m - 1$. In addition, ${\beta_\Phi}'$ is the only target of ${\beta_\Phi}_i^c$, and thus ${\beta_\Phi}_i^c$ will not contribute to the cut.

We will show that the rest of the nodes, i.e., nodes in V_2 that are associated with β_Φ, and the nodes in B, contribute nothing to the cut. For every $i \in [m]$, ${\beta_\Phi}_i^l \in S$ if and only if either $\beta_\Phi \vDash C_i$ or $\gamma_\Phi \vDash C_i$. It always happens that ${\beta_\Phi}_i^c \in S$, and ${\beta_\Phi}_i^r \in T$ if and only if

$\gamma_\Phi \nvDash C_i$, but in such case it must be that $\beta_\Phi \vDash C_i$, which implies that the edge $(\beta_{\Phi_i}^{\ l}, \beta_{\Phi_i}^{\ r})$ is not in the graph, thus the total contribution of $\beta_{\Phi_i}^{\ l}$ is zero.

For every $i \in [m]$, it is easy to verify that each of the following four implies the rest. $\beta_{\Phi_i}^{\ r} \in S$, $\gamma_\Phi \vDash C_i$, $C_i \in S$, and the edge (C_i, γ_Φ) is not in the graph. In the case where C_i and $\beta_{\Phi_i}^{\ r}$ are in T it is clear that they do not contribute anything, so we will focus on the other case. Since C_i is the only target of $\beta_{\Phi_i}^{\ r}$, $\beta_{\Phi_i}^{\ r}$ will not increase the cut capacity. In addition, since the edge (C_i, γ_Φ) is not in the graph, C_i does not increase the capacity of the cut either. Altogether we have bounded the total capacity of the cut by $2^{n/3}m - 1$, finishing the proof of Claim 3.2. ◀

Proceeding with the proof of Lemma 3.1, we now focus on the second direction. Assume that every assignment to F satisfies at most $p-1$ clauses. We remind that we need to prove that the maximum flow from every α to every γ is at least $2^{n/3}m$, and to do this we first fix α and γ. By the assumption, for every β there exist $m-(p-1) = m-p+1$ i's, such that α, β, and γ do not satisfy C_i, and we denote a set with this amount of such i's by I_β. Each of these i's induces a blue path $(\alpha \to C_i^{\nvDash} \to \beta_i^l \to \beta_i^r \to C_i \to \gamma)$ from α to γ, and so we pass a unit of flow through every one of them according to I_β, and for all β, in what we call the first phase. We note that so far, the flow sums up to $2^{n/3}(m-p+1)$, and so we carry on with shipping the second phase of flow through paths that are not entirely blue.

We claim that for every β, we can pass an additional amount of $m-(m-p+1) = p-1$ units through β', which would add up to a total flow of $2^{n/3}(m-p+1)+2^{n/3}(p-1) = 2^{n/3}m$, concluding the proof. Indeed, for every β, we ship flow in the following way. For every $i \in [m] \setminus I_\beta$, if $\alpha \nvDash C_i$ then send a unit through $(\alpha \to C_i^{\nvDash} \to \beta_i^l \to \beta_i^c \to \beta' \to v_B \to \gamma)$, and otherwise send a unit through $(\alpha \to C_i^{\vDash} \to \beta_i^c \to \beta' \to v_B \to \gamma)$.

Since we defined the flow in paths, we only need to show that the capacity constraints are satisfied, starting with edges of color blue. Edges of the forms (β_i^l, β_i^r), (β_i^r, C_i), and (C_i, γ) are only used in the first phase, where the flow in the first two is uniquely determined by β and $i \in I_\beta$, and so at most 1 unit of flow is passed through them, and the flow in the latter kind is determined by $i \in I_\beta$, and the same $i \in I_\beta$ can have at most $|\{\beta_i^r\}_\beta| = 2^{n/3}$ units of flow passing in (C_i, γ), and so the flow in it is also bounded. The flow in edges of the form $(C_i^{\nvDash}, \beta_i^l)$ in the first phase is uniquely determined by β and $i \in I_\beta$, and in the second phase uniquely according to β and $i \in [m] \setminus I_\beta$, and so will not be used twice, and the flow in edges of the form $(\alpha, C_i^{\nvDash})$ is determined in the first phase by $i \in I_\beta$ and in the second phase by $i \in [m] \setminus I_\beta$, and so will be used at most $\sum_\beta |I_\beta \cap \{i\}| + \sum_\beta |([m] \setminus I_\beta) \cap \{i\}| \leq 2^{n/3}$ times.

We now proceed to prove that red edges too do not have more flow than their capacity, and for this we only need to consider the second phase. Edges of the forms $(C_i^{\vDash}, \beta_i^c)$, (β_i^l, β_i^c), and (β_i^c, β') has flow that is uniquely determined by β and $i \in [m] \setminus I_\beta$ and so are not used more than once, edges of the form (β', v_B) has flow that is determined by β and thus have flow $|\{\beta_i^c\}_{i \in [m] \setminus I_\beta}| = |[m] \setminus I_\beta| = p-1$ and hence are properly bounded, and edges of the form (v_B, γ) have flow of size $(p-1)|\{\beta'\}_\beta|2^{n/3} = (p-1)2^{n/3}$. Finally, edges of the form $(\alpha, C_i^{\vDash})$ have flow that is determined by $i \in [m] \setminus I_\beta$ and so are used at most $|\{\beta_i^c\}_\beta| = 2^{n/3}$ times. Altogether, we have bounded the total flow in all the edges that were used in both phases, and so the capacity requirements follow, which completes the proof of the second direction and of Lemma 3.1. ◀

Proof of Theorem 1.6. We use Lemma 3.1 in the following way. Assume that for some $|S'|$ and $|T'|$ there is an algorithm for ST-Max-Flow that runs in time $O((|S'||T'|m)^{1-\varepsilon})$, and consider the version of ST-Max-Flow with S'' and T'' such that $S'' \cap T'' = \emptyset$. Applying such an algorithm repeatedly with S' and T' iterate over respective partitions of S'' and T'' of sizes

$|S''|/|S'|$ and $|T''|/|T'|$, respectively, would solve S'' and T'''s version of ST-Max-Flow and take time $(|S''|/|S'|)(|T''|/|T'|)O(|S'||T'|m)^{1-\varepsilon}) = O((|S''||T''|m)^{1-\varepsilon'})$ for some $\varepsilon' = \varepsilon'(\varepsilon)$, and Theorem 1.6 follows. ◀

4 Global Max-Flow

Proof of Theorem 1.10. Let U and V be an instance of the Orthogonal Vectors problem, where $|U| = |V| = n'$, and all the vectors are from $\{0,1\}^d$ for some $d = \mathsf{polylog}(n')$. We construct a graph $G = (U', V', D)$ as follows. U' contains a node u for every vector $u \in U$, V' contains a node v for every $v \in V$, and D contains three nodes $c_{0,0}$, $c_{0,1}$, and $c_{1,0}$ for every coordinate $c \in [d]$. For every $u \in U'$ and $c \in D$, we add an edge from u to $c_{0,0}$ and $c_{0,1}$ if $u[c] = 0$, and an edge from u to $c_{1,0}$ otherwise. Similarly, for every $v \in U'$ and $c \in D$, we add an edge from v to $c_{0,0}$ and $c_{1,0}$ if $v[c] = 0$, and an edge from v to $c_{0,1}$ otherwise. This graph has $n = n' + n' + 3d = O(n')$ nodes and at most $n' \cdot 2d + n' \cdot 2d = \tilde{O}(n')$ edges. For every $u \in U'$, $v \in V'$, and $c \in [d]$, there is exactly one path (of length 2) from u to v through nodes associated with c if and only if $u[c] \cdot v[c] = 0$, and no paths through them otherwise. Hence, the number of edge disjoint paths from u to v is d if their inner product is 0, and less than d otherwise, and so an algorithm for Maximum Local Edge Connectivity with strictly subquadratic runtime implies an algorithm for the Orthogonal Vectors problem with similar runtime, completing the proof. ◀

Acknowledgements. We thank Rajesh Chitnis and Bundit Laekhanukit for some useful conversations, and for their part in achieving the result on Global Max-Flow.

References

1 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based lower bounds for subset sum and bicriteria path. *CoRR*, 2017. URL: `http://arxiv.org/abs/1704.04546`.

2 Amir Abboud, Virginia Vassilevska-Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC'15, pages 41–50. ACM, 2015. `doi:10.1145/2746539.2746594`.

3 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows – theory, algorithms and applications*. Prentice Hall, 1993.

4 Srinivasa R. Arikati, Shiva Chaudhuri, and Christos D. Zaroliagis. All-pairs min-cut in sparse networks. *J. Algorithms*, 29(1):82–110, 1998. `doi:10.1006/jagm.1998.0961`.

5 Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012. `doi:10.4086/toc.2012.v008a006`.

6 Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, Inc., Hoboken, NJ, fourth edition, 2010.

7 Anand Bhalgat, Ramesh Hariharan, Telikepalli Kavitha, and Debmalya Panigrahi. An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs. In *39th Annual ACM Symposium on Theory of Computing*, STOC'07, pages 605–614. ACM, 2007. `doi:10.1145/1250790.1250879`.

8 Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International*

Proceedings in Informatics (LIPIcs), pages 22:1–22:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.SoCG.2016.22.

9 Glencora Borradaile and Philip Klein. An $\tilde{O}(n \log n)$ algorithm for maximum st-flow in a directed planar graph. *J. ACM*, 56(2):9:1–9:30, 2009. doi:10.1145/1502793.1502798.

10 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS'16, pages 261–270. ACM, 2016. doi:10.1145/2840728.2840746.

11 Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Graph connectivities, network coding, and expander graphs. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 190–199. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.55.

12 L. R. Ford, Jr. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

13 Greg N. Frederickson. Using cellular graph embeddings in solving all pairs shortest paths problems. *J. Algorithms*, 19(1):45–85, 1995. doi:10.1006/jagm.1995.1027.

14 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9:551–570, 1961. doi:10.1137/0109047.

15 Dan Gusfield. Very simple methods for all pairs network flow analysis. *SIAM J. Comput.*, 19(1):143–155, 1990. doi:10.1137/0219009.

16 Jianxiu Hao and James B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424–446, 1994. doi:10.1006/jagm.1994.1043.

17 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

18 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

19 Jakub Lacki, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Single source – all sinks max flows in planar digraphs. In *Proceedings of the 53rd IEEE Annual Symposium on Foundations of Computer Science*, FOCS'12, pages 599–608. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.66.

20 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{\text{rank}})$ iterations and faster algorithms for maximum flow. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS'14, pages 424–433. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.52.

21 Aleksander Mądry. Computing maximum flow with augmenting electrical flows. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science*, FOCS'16, pages 593–602. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.70.

22 Alexander Schrijver. On the history of the transportation and maximum flow problems. *Math. Program.*, 91(3):437–445, 2002. doi:10.1007/s101070100259.

23 Virginia Vassilevska-Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk). In *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*, volume 43 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17–29. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

24 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

On the Fine-Grained Complexity of One-Dimensional Dynamic Programming*†

Marvin Künnemann[1], Ramamohan Paturi[2], and Stefan Schneider[3]

1 **University of California, San Diego, CA, USA**
mkuennemann@eng.ucsd.edu
2 **University of California, San Diego, CA, USA**
paturi)@eng.ucsd.edu
3 **University of California, San Diego, CA, USA**
stschnei@eng.ucsd.edu

Abstract

In this paper, we investigate the complexity of *one-dimensional dynamic programming*, or more specifically, of the Least-Weight Subsequence (LWS) problem: Given a sequence of n data items together with weights for every pair of the items, the task is to determine a subsequence S minimizing the total weight of the pairs adjacent in S. A large number of natural problems can be formulated as LWS problems, yielding obvious $\mathcal{O}(n^2)$-time solutions.

In many interesting instances, the $\mathcal{O}(n^2)$-many weights can be succinctly represented. Yet except for near-linear time algorithms for some specific special cases, little is known about when an LWS instantiation admits a subquadratic-time algorithm and when it does not. In particular, no lower bounds for LWS instantiations have been known before. In an attempt to remedy this situation, we provide a general approach to study the fine-grained complexity of succinct instantiations of the LWS problem: Given an LWS instantiation we identify a highly parallel *core* problem that is subquadratically *equivalent*. This provides either an explanation for the apparent hardness of the problem or an avenue to find improved algorithms as the case may be.

More specifically, we prove subquadratic equivalences between the following pairs (an LWS instantiation and the corresponding core problem) of problems: a low-rank version of LWS and minimum inner product, finding the longest chain of nested boxes and vector domination, and a coin change problem which is closely related to the knapsack problem and $(\min, +)$-CONVOLUTION. Using these equivalences and known **SETH**-hardness results for some of the core problems, we deduce tight conditional lower bounds for the corresponding LWS instantiations. We also establish the $(\min, +)$-CONVOLUTION-hardness of the knapsack problem. Furthermore, we revisit some of the LWS instantiations which are known to be solvable in near-linear time and explain their easiness in terms of the easiness of the corresponding core problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Least-Weight Subsequence, SETH, Fine-Grained Complexity, Knapsack, Subquadratic Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.21

* A full version is available at [31].

† This research is supported by the Simons Foundation. This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 21; pp. 21:1–21:15

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

1 Introduction

Dynamic programming (DP) is one of the most fundamental paradigms for designing algorithms and a standard topic in textbooks on algorithms. Scientists from various disciplines have developed DP formulations for basic problems encountered in their applications. However, it is not clear whether the existing (often simple and straightforward) DP formulations are in fact optimal or nearly optimal. Our lack of understanding of the optimality of the DP formulations is particularly unsatisfactory since many of these problems are computational primitives.

Interestingly, there have been recent developments regarding the optimality of standard DP formulations for some specific problems, most importantly, conditional lower bounds assuming the Strong Exponential Time Hypothesis (**SETH**) [26]. Let us consider the longest common subsequence (LCS) problem as an illustrative example. It is defined as follows: Given two strings x and y of length at most n, compute the length of the longest string z that is a subsequence of both x and y. The standard DP formulation for the LCS problem involves computing a two-dimensional table requiring $\mathcal{O}(n^2)$ steps. This algorithm is slower than the fastest known algorithm due to Masek and Paterson [33] only by a polylogarithmic factor. However, there has been no progress in finding more efficient algorithms for this problem since the 1980s, which prompted attempts as early as in 1976 [5] to understand the barriers for efficient algorithms and to prove lower bounds. Unfortunately, there have not been any nontrivial unconditional lower bounds for this or any other problem in general models of computation. This state of affairs prompted researchers to consider *conditional* lower bounds based on conjectures such as 3-Sum conjecture [18] and more recently based on **ETH** [27] and **SETH** [26]. Researchers have found **ETH** and **SETH** to be useful to explain the exact complexity of several **NP**-complete problems (see the survey paper [32]). Surprisingly, Ryan Williams [39] has found a simple reduction from the CNF-SAT problem to the orthogonal vectors problem which under **SETH** leads to a matching quadratic lower bound for the orthogonal vectors problem. This in turn led to a number of conditional lower bound results for problems in **P** (including LCS and related problems) under **SETH** [6, 1, 10, 2, 22]. Also see [37] for a recent survey.

The DP formulation of the LCS problem is perhaps the conceptually simplest example of a *two-dimensional* DP formulation. In the standard formulation, each entry of an $n \times n$ table is computed in constant time. This property is typical for *alignment problems* which, for example, are used to model similarity between gene or protein sequences and for which LCS and Edit distance are the most prominent examples. Tight conditional lower bounds have recently been proved for a number of alignment problems [8, 6, 1, 10, 3].

In contrast, there are many problems for which natural quadratic-time DP formulations compute a *one-dimensional* table of length n by spending $\mathcal{O}(n)$-time per entry. The question arises: Can similar optimality results as for alignment problems be obtained for this fundamentally different setting? In pursuit of an answer, we investigate the optimality of one-dimensional DP formulations and obtain new (conditional) lower bounds which match the complexity of these standard DP formulations.

1-dimensional DP: The Least-Weight Subsequence (LWS) Problem. In this paper, we investigate the optimality of the standard DP formulation of the LWS problem. A classic example of an LWS problem is airplane refueling [24]: Given airport locations on a line, and a preferred distance per hop k (in miles), we define the penalty for flying k' miles as $(k - k')^2$. The goal is then to find a sequence of airports terminating at the last airport that minimizes the sum of the penalties. We now define the LWS problem formally.

▶ **Problem 1.1** (LWS). *We are given weights* $w_{i,j} \in \{-W, \dots, W\} \cup \{\infty\}$ *for every pair* $0 \le i < j \le n$ *and an arbitrary function* $g : \mathbb{Z} \to \mathbb{Z}$. *The* LWS *problem is to determine* $F[n]$ *which is defined by the following DP formulation.*

$$F[0] = 0,$$
$$F[j] = \min_{0 \le i < j} g(F[i]) + w_{i,j} \qquad \text{for } j = 1, \dots, n. \tag{1}$$

In the above definition, we did not specify the precise encoding of the problem. We typically consider *succinct instantiations* of LWS, where the input has subquadratic size (typically $\tilde{\mathcal{O}}(n)$) and the weights are a function of the input. In many cases, the input is a list of data items $x_0, \dots, x_n$ and $w_{i,j}$ is a function of x_i and x_j. For example, to formulate airplane refueling as an LWS problem, we let x_i be the location of the i'th airport, g be the identity function, and $w_{i,j} = (x_j - x_i - k)^2$.

The generality of the LWS definition captures a large variety of problems: it not only encompasses classical problems such as the pretty printing problem due to Knuth and Plass [30], the airplane refueling problem [24] and the longest increasing subsequence (LIS) [17], but also the unbounded subset sum problem [36, 9], a more general coin change problem that is effectively equivalent to the unbounded knapsack problem, 1-dimensional k-means clustering problem [23], finding longest R-chains (for an arbitrary binary relation R), and many others (for a more detailed overview and problem definitions, see the full version [31]).

Under mild assumptions on the encoding of the data items and weights, any instantiation of the LWS problems can be solved in time $\mathcal{O}(n^2)$ using (1) for determining the values $F[j], j = 1, \dots, n$ in time $\mathcal{O}(n)$ each. However, the best known algorithms for the LWS problems differ quite significantly in their time complexity. Some problems including the pretty printing problem, the airline refueling problem and LIS turn out to be solvable in near-linear time, while no subquadratic algorithms are known for the unbounded knapsack problem or for finding the longest R-chain.

The main goal of the paper is to investigate the optimality of the LWS DP formulation for various problems by proving conditional lower bounds.

Succinct LWS instantiations. In the extremely long presentation of an LWS problem, the weights $w_{i,j}$ are given explicitly. This is, however, not a very interesting case from a computational point of view, as the standard DP formulation takes linear time (in the size of the input) to compute $F[n]$. In the example of the airplane refueling problem, the size of the input is only $\mathcal{O}(n)$ assuming that the values of the data items are bounded by some polynomial in n. For such succinct representations, we ask if the quadratic-time algorithm based on the standard LWS DP formulation is optimal. Our approach is to study several natural succinct versions of the LWS problem (by specifying the type of data items and the weight function[1]) and determine their complexity.

Our Contributions and Results. The main contributions of our paper include a general framework for reducing succinct LWS instantiations to what we call the *core* problems and proving subquadratic equivalences between them. Such subquadratic equivalences are interesting for two reasons. First, they allow us to conclude conditional lower bounds for certain LWS instantiations, where previously no lower bounds are known. Second,

[1] In all our applications, the function g is the trivial identity function.

subquadratic (or more general fine-grained) equivalences are more useful since they let us translate *easiness* in addition to hardness results.

Our results include tight (up to subpolynomial factors) conditional lower bounds for several LWS instantiations with succinct representations. These instantiations include the coin change problem, low-rank versions of the LWS problem, and the longest subchain problems. Our results are somewhat more general. We propose a *factorization* of the LWS problem into a *core* problem and a fine-grained reduction from the LWS problem to the core problem. The idea is that core problems (which are often well-known problems) capture the hardness of the LWS problem and act as a potential barrier for more efficient algorithms. While we do not formally define the notion of a core problem, we identify several core problems which share several interesting properties. For example, they do not admit natural DP formulations and are easy to parallelize. In contrast, the quadratic-time DP formulation of LWS problems requires the entries $F[i]$ to be computed in order, suggesting that the general problem might be inherently sequential.

The reductions between LWS problems and core problems involve a natural intermediate problem, which we call the Static-LWS problem. We first reduce the LWS problem to the Static-LWS problem in a general way and then reduce the Static-LWS problem to a core problem. The first reduction is divide-and-conquer in nature and is inherently sequential. The latter reduction is specific to the instantiation of the LWS problem. The Static-LWS problem is easy to parallelize and does not have a natural DP formulation. However, the problem is not necessarily a natural problem. The Static-LWS problem can be thought of as a generic core problem, but it is output-intensive.

In the other direction, we show that many of the core problems can be reduced to the corresponding LWS instantiations thus establishing an equivalency between LWS instantiations and their core problems. This equivalence enables us to translate both the hardness and easiness results (i.e., the subquadratic-time algorithms) for the core problems to the corresponding LWS instantiations.

The first natural succinct representation of the LWS problem we consider is the low-rank LWS problem, where the weight matrix $\mathbf{W} = (w_{i,j})$ is of low rank and thus representable as $\mathbf{W} = L \cdot R$ where L and R^{T} are $(n \times n^{o(1)})$-matrices. For this low-rank LWS problem, we identify the minimum inner product problem (MinInnProd) as a suitable core problem. It is only natural and not particularly surprising that MinInnProd can be reduced to the low-rank LWS problem which shows the **SETH**-hardness of the low-rank LWS problem. The other direction is more surprising: Inspired by an elegant trick of Vassilevska Williams and Williams [40], we are able to show a subquadratic-time reduction from the (highly sequential) low-rank LWS problem to the (highly parallel) MinInnProd problem. Thus, the very compact problem MinInnProd problem captures exactly the complexity of the low-rank LWS problem (under subquadratic reductions).

We also show that the coin change problem is subquadratically equivalent to the $(\min, +)$-convolution problem. In the coin change problem, the weight matrix $\mathbf{W}$ is succinctly given as a Toeplitz matrix. At this point, the conditional hardness of the $(\min, +)$-convolution problem is unknown. Only very recently and independent of our work, a detailed treatment of Cygan et al. [13] considers quadratic-complexity of $(\min, +)$-convolution as a hardness assumption and discusses its relation to more established assumptions. The quadratic-time hardness of the $(\min, +)$-convolution problem would be very interesting, since it is known that the $(\min, +)$-convolution problem is reducible to the 3-Sum problem and the APSP problem (see also [13]). However, recent results give surprising subquadratic-time algorithms for special cases of $(\min, +)$-convolution [12]. If these subquadratic-time

Table 1 Summary of our results.

Name	Weights	Equivalent Core	Reference
Coin Change	Toeplitz matrix: $w_{i,j} = w_{j-i}$	$(\min,+)$-CONVOLUTION	Theorem 5.9
	Remark: Subquadratically equivalent to UNBOUNDEDKNAPSACK		
LowRankLWS	Low rank representation: $w_{i,j} = \langle \sigma_i, \mu_j \rangle$	MININNPROD	Theorem 4.7
R-chains	matrix induced by R: $w_{i,j} = w_j$ if $R(x_i, x_j)$ and ∞ o/w	SELECTION(R)	Theorem 6.3 Theorem 6.4
	Remark: Results below are corollaries.		
NestedBoxes	$w_{i,j} = -1$ if B_j contains B_i	VECTORDOMINATION	
SubsetChain	$w_{i,j} = -1$ if $S_i \subseteq S_j$	ORTHOGONALVECTORS	

algorithms extend to the general $(\min,+)$-CONVOLUTION problem, our equivalence result also provides a subquadratic-time algorithm for the coin change problem and the closely related unbounded knapsack problem. Our reductions also give, as a corollary, a quadratic-time $(\min,+)$-CONVOLUTION-based lower bound for the bounded case of knapsack. We remark that independently of our results, [13] gave randomized subquadratic equivalences of $(\min,+)$-CONVOLUTION to unbounded knapsack (while we give deterministic reductions) and bounded Knapsack (where we only give a $(\min,+)$-CONVOLUTION-based lower bound).

We next consider the problem of finding longest chains: here, we search for the longest subsequence (chain) in the input sequence such that all adjacent pairs in the subsequence are contained in some binary relation R. We show that for any binary relation R satisfying certain conditions the chaining problem is subquadratically equivalent to a corresponding (highly parallel) selection problem. As corollaries, we get equivalences between finding the longest chain of nested boxes (NESTEDBOXES) and VECTORDOMINATION as well as between finding the longest subset chain (SUBSETCHAIN) and the orthogonal vectors (OV) problem. Interestingly, these results have algorithmic implications: known algorithms for low-dimensional vector domination and low-dimensional orthogonal vectors translate to faster algorithms for low-dimensional NESTEDBOXES and SUBSETCHAIN for small universe size.

Table 1 lists the LWS succinct instantiations (as discussed above) and their corresponding core problems. For a detailed treatment of all LWS instantiations and core problems considered in this work, see the full version of this paper [31].

Finally, we revisit classic problems including the longest increasing subsequence problem, the unbounded subset sum problem and the concave LWS problem and analyze the STATIC-LWS instantiations to immediately infer that the corresponding core problem can be solved in near-linear time. Table 2 gives an overview of some of the problems we look at in this context.

Related Work. LWS has been introduced by Hirschberg and Larmore [24]. If the weight function satisfies the *quadrangle inequality* formalized by Yao [41], one obtains the *concave LWS* problem (CONCLWS), for which they give an $\mathcal{O}(n \log n)$-time algorithm. Subsequently, improved algorithms solving CONCLWS in time $\mathcal{O}(n)$ were given [38, 20]. This yields a fairly large class of weight functions (including, e.g., the pretty printing and airplane refueling problems) for which linear-time solutions exist. To generalize this class of problems, further

Table 2 Near-linear time algorithms following from the proposed framework.

Name	Weights	$\tilde{\mathcal{O}}(n)$-algorithm via	Reference
Longest Increasing Subsequence	matrix induced by $R_<$: $w_{i,j} = -1$ if $x_i < x_j$	SORTING	[17], full version [31]
Unbounded Subset Sum	Toeplitz $\{0, \infty\}$ matrix: $w_{i,j} = w_{j-i} \in \{0, \infty\}$	CONVOLUTION	[9], full version [31]
Concave 1-dim. DP	concave matrix: $w_{i,j} + w_{i',j'} \leq w_{i',j} + w_{i,j'}$ for $i \leq i' \leq j \leq j'$	SMAWK problem	[24, 20, 38], full version [31]

works address convex weight functions[2] [19, 35, 29] as well as certain combinations of convex and concave weight functions [15] and provide near-linear time algorithms. For a more comprehensive overview over these algorithms and further applications of the LWS problem, we refer the reader to Eppstein's PhD thesis [16].

Apart from these notions of concavity and convexity, results on succinct LWS problems are typically more scattered and problem-specific (see, e.g., [17, 30, 9, 23]; furthermore, a closely related recurrence to (1) pops up when solving bitonic TSP [14]). An exception to this rule is a study of the parallel complexity of LWS [21].

Organization. After setting up notation and conventions in Section 2, Section 3 gives a general reduction from LWS instantiations to STATIC-LWS that is independent of the representation of the weight matrix. Section 4 contains the result on low-rank LWS. Section 5 proves the subquadratic equivalence of the coin change problem and $(\min, +)$-CONVOLUTION, while Section 6 discusses chaining problems and their corresponding selection (core) problem. Due to space constraints, most proofs and our discussion of near-linear time algorithms are deferred to the full version of this article [31].

2 Preliminaries

In this section, we state our notational conventions and list the main problems considered in this work.

Notation and Conventions. Problem A *subquadratically reduces* to problem B, denoted $A \leq_2 B$, if for any $\varepsilon > 0$ there is a $\delta > 0$ such that the existence of a $\mathcal{O}(n^{2-\varepsilon})$-time algorithm for B implies a $\mathcal{O}(n^{2-\delta})$-time algorithm for A. We call the two problems subquadratically equivalent, denoted $A \equiv_2 B$, if there are subquadratic reductions both ways.

We let $[n] := \{1, \ldots, n\}$. When stating running time, we use the notation $\tilde{\mathcal{O}}(\cdot)$ to hide polylogarithmic factors. For a problem P, we write T^P for its time complexity. We generally assume the word-RAM model of computation with word size $w = \Theta(\log n)$. For most problems defined in this paper, we consider inputs to be integers in the range $\{-W, \ldots, W\}$ where W fits in a constant number of words[3]. For vectors, we use d for the dimension and generally assume $d = n^{o(1)}$.

[2] A weight function is convex if it satisfies the inverse of the quadrangle inequality.

[3] For the purposes of our reductions, even values up to $W = 2^{n^{o(1)}}$ would be fine.

Succinct LWS Instantiations. In the definition of LWS (Problem 1.1) we did not fix the encoding of the problem (in particular the representation of the weights $w_{i,j}$ and the function g). Assuming that g and the weights can be determined in $\tilde{\mathcal{O}}(1)$ and that $W = \texttt{poly}(n)$, this problem can naturally be solved in time $\tilde{\mathcal{O}}(n^2)$, by evaluating the central recurrence (1) for each $j = 1, \ldots, n$ – this takes $\tilde{\mathcal{O}}(n)$ time for each j, since we take the minimum over at most n expressions that can be evaluated in time $\tilde{\mathcal{O}}(1)$ by accessing the previously computed entries $F[0], \ldots, F[j-1]$ as well as computing g. We assume from now on that g is the identity function, as this is the case for all our applications. Thus it suffices to define the type of data items and the corresponding weight matrix to specify an LWS instantiation. Throughout this paper, whenever we fix a representation of the weight matrix $\mathbf{W} = (w_{i,j})_{i,j}$, we denote the corresponding problem LWS($\mathbf{W}$).

3 Static LWS

Our reductions from LWS instantiations to core problems go through intermediate problems that share some of the characteristics of core problems, as well as some of the characteristics of LWS. In particular, these problems are naturally parallelizable and their brute-force algorithm is already quadratic time, similar to core problems. On the other hand their definitions are closely related to the definition of LWS. Other than core problems, our intermediate problems are not decision problems but ask to compute some linear sized output. Towards making this notion more precise, we define a generic intermediate problem called Static-LWS.

▶ **Problem 3.1** (Static-LWS($\mathbf{W}$)). *Fix an instance of* LWS($\mathbf{W}$). *Given intervals of indices* $I := \{a+1, \ldots, a+N\}$ *and* $J := \{a+N+1, \ldots, a+2N\}$ *with* a, N *such that* $I, J \subseteq [n]$, *together with the values* $F[a+1], \ldots, F[a+N]$, *the Static Least-Weight Subsequence Problem (*Static-LWS*) asks to determine*

$$F'[j] := \min_{i \in I} F[i] + w_{i,j} \qquad \text{for all } j \in J.$$

The main purpose of this section is to give a reduction from LWS($\mathbf{W}$) to Static-LWS($\mathbf{W}$) that is independent of the weight matrix $\mathbf{W}$ and therefore independent of the succinct LWS instantiations we consider throughout this paper. This reduction is a key step in our reductions from LWS to their corresponding core problems.

The reduction is a divide-and-conquer scheme that divides the LWS problem into two subproblems of half the size each and Static-LWS to combine the two. Crucially, the two subproblems have to be solved sequentially. The reduction therefore captures the sequential nature of the LWS problem, while Static-LWS captures a parallelizable part of the problem.

In a certain sense, this reduction has appeared implicitly in previous work on LWS [24]. In particular, the reduction of ConcLWS to the SMAWK problem by Galil and Park [20] can be thought of as a variant of this reduction specialized to the concave case to avoid log-factors.

▶ **Lemma 3.2** (LWS($\mathbf{W}$) $\leq_2$ Static-LWS($\mathbf{W}$)). *For any choice of* $\mathbf{W}$, *if* Static-LWS($\mathbf{W}$) *can be solved in time* $\mathcal{O}(N^{2-\varepsilon})$ *for some* $\varepsilon > 0$, *then* LWS($\mathbf{W}$) *can be solved in time* $\tilde{\mathcal{O}}(n^{2-\varepsilon})$.

Proof. In what follows, we fix LWS as LWS($\mathbf{W}$) and Static-LWS as Static-LWS($\mathbf{W}$).

We define the subproblem $S(\{i, \ldots, j\}, (f_i, \ldots, f_j))$ that given an interval spanned by $1 \leq i \leq j \leq n$ and values $f_k = \min_{0 \leq k' < i} F[k'] + w_{k',k}$ for each point $k \in \{i, \ldots, j\}$, computes all values $F[k]$ for $k \in \{i, \ldots, j\}$. Note that a call to $S([n], (w_{0,1}, \ldots, w_{0,n}))$ solves the LWS problem, since $F[0] = 0$ and thus the values of $f_k, k \in [n]$ are correctly initialized.

Algorithm 1 Reducing LWS to Static-LWS

1: **function** $S(\{i,\dots,j\},(f_i,\dots,f_j))$
2: **if** $i=j$ **then**
3: **return** $F[i]\leftarrow f_i$
4: $m\leftarrow\lceil\frac{j-i}{2}\rceil$
5: $(F[i],\dots,F[i+m-1])\leftarrow S(\{i,\dots,i+m-1\},(f_i,\dots,f_{i+m-1}))$
6: solve Static-LWS on the subinstance given by $I:=\{i,\dots,i+m-1\}$ and $J:=\{i+m,\dots,i+2m-1\}$:
7: ▷ obtains values $F'[k]=\min_{i\le k'<i+m}F[k']+w_{k',k}$ for $k=i+m,\dots,i+2m-1$.
8: $f'_k\leftarrow\min\{f_k,F'[k]\}$ for all $k=i+m,\dots,i+2m-1$.
9: $(F[i+m],\dots,F[i+2m-1])\leftarrow S(\{i+m,\dots,i+2m-1\},(f'_{i+m},\dots,f'_{i+2m-1}))$
10: **if** $j=i+2m$ **then**
11: $F[j]:=\min\{f_j,\min_{i\le k<j}F[k]+w_{k,j}\}$.
12: **return** $(F[i],\dots,F[j])$

We solve S using Algorithm 1.

We briefly argue correctness, using the invariant that $f_k=\min_{0\le k'<i}F[k']+w_{k',k}$ in every call to S. If S is called with $i=j$, then the invariant yields $f_i=\min_{0\le k'<i}F[k']+w_{k',i}=F[i]$, thus $F[i]$ is computed correctly. For the call in Line 5, the invariant is fulfilled by assumption, hence the values $(F[i],\dots,F[i+m-1])$ are correctly computed. For the call in Line 9, we note that for $k=i+m,\dots,i+2m-1$, we have that f'_k equals

$$\min\{f_k,F'[k]\}=\min\{\min_{0\le k'<i}F[k']+w_{k',k},\min_{i\le k'<i+m}F[k']+w_{k',k}\}=\min_{0\le k'<i+m}F[k']+w_{k',k}.$$

Hence the invariant remains satisfied. Thus, the values $(F[i+m],\dots,F[i+2m-1])$ are correctly computed. Finally, if $j=i+2m$, we compute the remaining value $F[j]$ correctly, since $f_j=\min_{0\le k<i}F[k]+w_{k,j}$ by assumption.

To analyze the running time $T^S(n)$ of S on an interval of length $n:=j-i+1$, note that each call results in two recursive calls of interval lengths at most $n/2$. In each call, we need an additional overhead that is linear in n and $T^{\text{Static-LWS}}(n/2)$. Solving the corresponding recursion $T^S(n)\le 2T^S(n/2)+T^{\text{Static-LWS}}(n/2)+\mathcal{O}(n)$, we obtain that an $\mathcal{O}(N^{2-\varepsilon})$-time algorithm Static-LWS, with $0<\varepsilon<1$ yields $T^{\text{LWS}}(n)\le T^S(n)=\mathcal{O}(n^{2-\varepsilon})$. Similarly, an $\mathcal{O}(N\log^c N)$-time algorithm for Static-LWS would result in an $\mathcal{O}(n\log^{c+1}n)$-time algorithm for LWS. ◀

4 LowRankLWS

In this section we prove the first equivalence between an instantiation of LWS and a core problem. Specifically, we first analyze the following canonical succinct representation of a low-rank weight matrix $\mathbf{W}=(w_{i,j})_{i,j}$: If $\mathbf{W}$ is of rank $d\ll n$, we can write it more succinctly as $\mathbf{W}=L\cdot R$, where L and R are $(n\times d)$- and $(d\times n)$ matrices, respectively. We can express the resulting natural LWS problem equivalently as follows.

▶ **Problem 4.1** (LowRankLWS). *We define the* LWS *instantiation* LowRankLWS $=$ LWS($\mathbf{W}_{\text{LowRank}}$) *as follows.*
Data: *out-vectors* $\mu_0,\dots,\mu_{n-1}\in\{-W,\dots,W\}^d$, *in-vectors* $\sigma_1,\dots,\sigma_n\in\{-W,\dots,W\}^d$
Weights: $w(i,j)=\langle\mu_i,\sigma_j\rangle$ *for* $0\le i<j\le n$

In this section, we show that this problem is equivalent, under subquadratic reductions, to the following *non-sequential* problem.

▶ **Problem 4.2** (MinInnProd). *Given $a_1, \dots, a_n, b_1, \dots, b_n \in \{-W, \dots, W\}^d$ and a natural number $r \in \mathbb{Z}$, determine if there is a pair i, j satisfying $\langle a_i, b_j \rangle \leq r$.*

This is interesting for a number of reasons. For one, MinInnProd is a fairly natural problem and, as opposed to LowRankLWS it is not inherently sequential in its definition. We understand MinInnProd comparably well both from an upper and from a lower bound perspective. Using ray shooting data structures [34] we can solve MinInnProd in strongly subquadratic time if d is constant. At the same time, if $d = \omega(\log n)$, the problem is quadratic-time **SETH**-hard. By showing subquadratic equivalence between MinInnProd and LowRankLWS, we can conclude both these results, as well as any future improvements, for LowRankLWS.

There is a simple reduction from MinInnProd to LowRankLWS that along the way proves quadratic-time **SETH**-hardness of LowRankLWS.

▶ **Lemma 4.3.** *It holds that $T^{\text{MinInnProd}}(n, d, W) \leq T^{\text{LowRankLWS}}(2n+1, d+2, dW) + \mathcal{O}(nd)$.*

To prove the other direction, we will use the quite general approach to compute the sequential LWS problem by reducing to Static-LWS (Lemma 3.2). In particular, for the special case of LowRankLWS, it is not difficult to see that its static version boils down to the following natural reformulation.

▶ **Problem 4.4** (AllInnProd). *Given vectors $a_1, \dots, a_n \in \{-W, \dots, W\}^d$ and $b_1, \dots, b_n \in \{-W, \dots, W\}^d$, determine* for all $j \in [n]$, *the value* $\min_{i \in [n]} \langle a_i, b_j \rangle$.

▶ **Lemma 4.5** (Static-LWS($\mathbf{W}_{\text{LowRank}}$) $\leq_2$ AllInnProd). *We have*

$$T^{\text{Static-LWS}(\mathbf{W}_{\text{LowRank}})}(n, d, W) \leq T^{\text{AllInnProd}}(n, d+1, nW) + \mathcal{O}(nd).$$

Finally, inspired by an elegant trick of [40], we reduce AllInnProd to MinInnProd.

▶ **Lemma 4.6** (AllInnProd $\leq_2$ MinInnProd). *We have*

$$T^{\text{AllInnProd}}(n, d, W) \leq \mathcal{O}(n \cdot T^{\text{MinInnProd}}(\sqrt{n}, d+3, ndW^2) \cdot \log^2 nW).$$

By the sequence of lemmas above and Lemma 3.2, we obtain our subquadratic equivalence of LowRankLWS to its core problem.

▶ **Theorem 4.7.** *We have* LowRankLWS $\equiv_2$ MinInnProd.

5 Coin Change and Knapsack Problems

In this section, we focus on the following problem related to Knapsack: Assume we are given coins of denominations $d_1, \dots, d_m$ with corresponding weights $w_1, \dots, w_m$ and a target value n, determine a way to represent n using these coins (where each coin can be used arbitrarily often) minimizing the total sum of weights of the coins used. Since without loss of generality $d_i \leq n$ for all i, we can assume that $m \leq n$ and think of n as our problem size. In particular, we describe the input by weights $w_1, \dots, w_n$ where w_i denotes the weight of the coin of denomination i (if no coin with denomination i exists, we set $w_i = \infty$). It is straightforward to see that this problem is an LWS instance LWS($\mathbf{W}_{\text{cc}}$), where the weight matrix $\mathbf{W}_{\text{cc}}$ is a Toeplitz matrix.

▶ **Problem 5.1** (CC). *We define the following* LWS *instantiation* $\mathrm{CC} = \mathrm{LWS}(\mathbf{W}_{\mathrm{cc}})$.
Data: *weight sequence* $w = (w_1, \ldots, w_n)$ *with* $w_i \in \{-W, \ldots, W\} \cup \{\infty\}$
Weights: $w_{i,j} = w_{j-i}$ *for* $0 \leq i < j \leq n$

Translated into a Knapsack-type formulation (i.e., denominations are weights, weights are profits, and the objective becomes to maximize the profit), the problem differs from UnboundedKnapsack only in that it searches for the most profitable multiset of items of weight *exactly* n, instead of *at most* n.

▶ **Problem 5.2** (UnboundedKnapsack). *We are given a sequence of profits* $p = (p_1, \ldots, p_n)$ *with* $p_i \in \{0, 1, \ldots, W\}$*, that is, the item of size* i *has profit* p_i*. Find the total profit of the multiset of indices* I *such that* $\sum_{i \in I} i \leq n$ *and the total profit* $\sum_{i \in I} p_i$ *is maximized.*

The purpose of this section is to show that both CC and UnboundedKnapsack are subquadratically equivalent to the $(\min, +)$-convolution problem. Along the way, we also prove quadratic-time $(\min, +)$-convolution-hardness of Knapsack. Recall the definition of $(\min, +)$-convolution.

▶ **Problem 5.3** ($(\min, +)$-convolution). *Given* n*-dimensional vectors* $a = (a_0, \ldots, a_{n-1})$*,* $b = (b_0, \ldots, b_{n-1}) \in \{-W, \ldots, W\}^n$*, determine its* $(\min, +)$-convolution $a * b$ *defined by*

$$(a * b)_k = \min_{0 \leq i,j < n : i+j=k} a_i + b_j \qquad \text{for all } 0 \leq k \leq 2n-2.$$

As opposed to the classical convolution, solvable in time $\mathcal{O}(n \log n)$ using FFT, no strongly subquadratic algorithm for $(\min, +)$-convolution is known. Compared to the popular orthogonal vectors problem, we have less support for believing that no $\mathcal{O}(n^{2-\varepsilon})$-time algorithm for $(\min, +)$-convolution exists. In particular, interesting special cases can be solved in subquadratic time [12] and there are subquadratic-time co-nondeterministic and nondeterministic algorithms [7, 11]. At the same time, breaking this long-standing quadratic-time barrier is a prerequisite for progress on refuting the 3SUM and APSP conjectures (see also [13]). This makes it an interesting target particularly for proving subquadratic *equivalences*, since both positive and negative resolutions of this open question appear to be reasonable possibilities.

To obtain our result, we address two issues: (1) We show an equivalence between the problem of determining only the value $F[n]$, i.e., the best way to give change only for the target value n, and to determine *all values* $F[1], \ldots, F[n]$, which we call the *output-intensive version*. (2) We show that the output-intensive version is subquadratic equivalent to $(\min, +)$-convolution.

▶ **Problem 5.4** (oiCC). *The output-intensive version of* CC *is to determine, given an input to* CC*, all values* $F[1], \ldots, F[n]$.

We first consider issue (2) and prove $(\min, +)$-convolution-hardness of oiCC.

▶ **Lemma 5.5** ($(\min, +)\mathrm{conv} \leq_2$ oiCC). *We have* $T^{(\min,+)\mathrm{conv}}(n, W) \leq T^{\mathrm{oiCC}}(6n, 4(2W + 1)) + \mathcal{O}(n)$.

Using the notion of Static-LWS, the other direction is straight-forward.

▶ **Lemma 5.6.** *We have* $\mathrm{oiCC} \leq_2 \text{Static-LWS}(\mathbf{W}_{\mathrm{cc}}) \leq_2 (\min, +)\mathrm{conv}$.

The last two lemmas resolve issue (2). We proceed to issue (1) and show that the output-intensive version is subquadratically equivalent to both CC and UnboundedKnapsack that only ask to determine a single output number.

It is trivial to see that UnboundedKnapsack $\leq_2$ oiCC. Furthermore, there is a simple reduction from CC to UnboundedKnapsack.

▶ **Oberservation 5.7** (CC $\leq_2$ UnboundedKnapsack $\leq_2$ oiCC)**.** *We have* $T^{\mathrm{CC}}(n,W) \leq T^{\textsc{UnboundedKnapsack}}(n,nW) + \mathcal{O}(n)$ *and* $T^{\textsc{UnboundedKnapsack}}(n,W) \leq T^{\mathrm{oiCC}}(n,W) + \mathcal{O}(n)$.

The remaining part is similar in spirit to Lemma 4.6: Somewhat surprisingly, the same general approach works despite the much more sequential nature of Knapsack and CC – this sequentiality can be taken care of by a more careful treatment of appropriate subproblems that involves solving them in a particular order and feeding them with information gained during the process.

▶ **Lemma 5.8** (oiCC $\leq_2$ CC)**.** *We have* $T^{\mathrm{oiCC}}(n,W) \leq \mathcal{O}(\log(nW)\cdot n \cdot T^{\mathrm{CC}}(24\sqrt{n}, 3n^2W))$.

The lemmas above and their underlying reductions prove the following theorem.

▶ **Theorem 5.9.** *We have* $(\min,+)$conv $\equiv_2$ CC $\equiv_2$ UnboundedKnapsack*. Furthermore, the bounded version of* Knapsack *admits no strongly subquadratic-time algorithm unless* $(\min,+)$-convolution *can be solved in strongly subquadratic time.*

6 Chain LWS

In this section we consider a special case of Least-Weight Subsequence problems called the Chain Least-Weight Subsequence (ChainLWS) problem. This captures problems in which edge weights are given implicitly by a relation R that determines which pairs of data items we are allowed to chain. The aim is to find the longest chain.

An example of a Chain Least-Weight Subsequence problem is the NestedBoxes problem. Given n boxes in d dimensions, given as non-negative, d-dimensional vectors $b_1,\ldots,b_n$, find the longest chain such that each box fits into the next (without rotation). We say box that box a fits into box b if for all dimensions $1 \leq i \leq d$, $a_i \leq b_i$.

NestedBoxes is not immediately a Least-Weight Subsequence problem, as for Least-Weight subsequence problems we are given a sequence of data items, and require any sequence to start at the first item and end at the last. However, we can easily convert NestedBoxes into a LWS problem by sorting the vectors by the sum of the entries and introducing two special boxes, one very small box $\bot$ such that $\bot$ fits into any box b_i and one very large box $\top$ such that any b_i fits into $\top$.

We define the Chain Least-Weight Subsequence problem with respect to any relation R and consider a weighted version where data items are given weights. To make the definition consistent with the definition of LWS the output is the weight of the sequence that minimizes the sum of the weights.

▶ **Problem 6.1** (ChainLWS)**.** *Fix a set of objects* D *and a relation* $R \subseteq D \times D$*. We define the following* LWS *instantiation* $\textsc{ChainLWS}(R) = \mathrm{LWS}(\mathbf{W}_{\textsc{ChainLWS}(R)})$.
Data: *sequence of objects* $d_0,\ldots,d_n \in D$ *with weights* $w_1,\ldots,w_n \in \{-W,\ldots,W\}$.
Weights: $w_{i,j} = \begin{cases} w_j & \text{if } (x_i,x_j) \in R, \\ \infty & \text{otherwise,} \end{cases}$ *for* $0 \leq i < j \leq n$.

The input to the (weighted) Chain Least-Weight Subsequence problem is a *sequence* of data items, and not a set. Finding the longest chain in a *set* of data items is **NP**-complete in general. For example, consider the box overlap problem: The input is a set of boxes in two dimensions, given by the top left corner and the bottom right corner, and the relation

consists of all pairs such that the two boxes overlap. This problem is a generalization of the Hamiltonian path problem on induced subgraphs of the two-dimensional grid, which is an **NP**-complete problem [28].

We relate ChainLWS(R) to the class of *selection* problems with respect to the same relation R.

▶ **Problem 6.2** (Selection Problem). *Let D be a set of objects, let $R \subseteq D \times D$ be a relation and let $D_1, D_2 \subseteq D^n$. Given two sequences of inputs $(a_1, \ldots, a_n) \in D_1$ and $(b_1, \ldots, b_n) \in D_2$, determine if there is i, j satisfying $R(a_i, b_j)$. We denote this selection problem with respect to the relation R and sets D_1, D_2 by* Selection(R^{D_1,D_2}). *If $D_1 = D_2 = D^n$, we denote the problem by* Selection(R).

The class of selection problems includes several well-studied problems including MinInnProd, OV [39, 4] and VectorDomination [25].

We give a subquadratic reduction from ChainLWS(R) to Selection(R), independently of R. The proof is again based on Static-LWS and a variation on a trick of [40].

▶ **Theorem 6.3.** *For all relations R such that R can be computed in time subpolynomial in the number of data items n,* ChainLWS(R) $\leq_2$ Selection(R).

For the other direction, we do not have a reduction that is independent of the relation R. Instead, we give sufficient conditions for the existence of such subquadratic reductions.

▶ **Theorem 6.4.** *Let D be a set of objects and $D_1, D_2 \subseteq D^n$ be a set of possible sequences. Consider any relation $R \subseteq D \times D$ satisfying the following properties.*

- *There is a data item $\perp$ such that $(\perp, d) \in R$ for all $d \in D$.*
- *There is a data item $\top$ such that $(d, \top) \in R$ for all $d \in D$.*
- *For all $a \in \{1, 2\}$ and any set of data items $(d_1, \ldots, d_n) \in D_a$ there is a permutation of indices $i_1, \ldots, i_n$ such that for any $j < k$, $(d_{i_j}, d_{i_k}) \notin R$. This ordering can be computed in time $\mathcal{O}(n^{2-\delta})$ for $\delta > 0$. We call this ordering the natural ordering.*

Then Selection(R^{D_1,D_2}) $\leq_2$ ChainLWS(R).

We call a relation satisfying the conditions above a *topological* relation. An immediate corollary is that if we can subquadratically reduce Selection(R) to Selection(R') for some topological relation R', then Selection(R) $\leq_2$ ChainLWS(R').

We conclude by providing interesting instantiations of the subquadratic equivalence of Selection and ChainLWS.

▶ **Corollary 6.5** (NestedBoxes $\equiv_2$ VectorDomination). *The weighted* NestedBoxes *problem on $d = c \log n$ dimensions can be solved in time $n^{2-(1/\mathcal{O}(c \log^2 c))}$. For $d = \omega(\log n)$, the (unweighted)* NestedBoxes *problem cannot be solved in time $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$ assuming* **SETH**.

If we restrict NestedBoxes and VectorDomination to Boolean vectors, then we get SubsetChain and SetContainment, respectively. In this case the upper bound improves to $n^{2-1/\mathcal{O}(\log c)}$ [4]. Note that SetContainment $\equiv_2$ OV, hence SubsetChain $\equiv_2$ OV.

7 Open Problems

We discuss the complexity of some succinct LWS instantiations both from an upper bound and a lower bound perspective by proving equivalences with a number of comparably well-studied core problems. The succinct instantiations we study include natural problems

such as LowRankLWS, CC, ChainLWS including NestedBoxes and SubsetChain, as well as previously studied instantiations such as ConcLWS and LIS. A number of open questions remain. Our results do not generalize to arbitrary instantiations of LWS. In particular, Static-LWS does not seem to reduce subquadratically to the problem of finding the minimum element in a succinctly descibed matrix. With LowRankLWS and CC we do provide instances for which we can identify equivalent core problems, and it will be interesting to find further examples or even sufficient conditions for which we can reduce LWS to other problems and vice versa.

For the case of ChainLWS, we are able to generalize the reduction from LWS to Selection problems. However, the reduction, while preserving subquadratic algorithms, does not preserve near-linear time algorithms. For some cases, such as LIS, we are able to reconstruct a near-linear time algorithm, which raises the question of what conditions are necessary to do that. Similarly, we give sufficient conditions to reduce from Selection to ChainLWS, and other sufficient or even necessary conditions should be explored for both black-box as well as white-box reductions.

Acknowledgments. We would like to thank Karl Bringmann and Russell Impagliazzo for helpful discussions and comments.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 59–78, 2015.

2 Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with Edit Distance and friends or: A polylog shaved is a lower bound made. In *Proc. 48th Annual ACM Symposium on Symposium on Theory of Computing (STOC'16)*, 2016. To appear.

3 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, pages 39–51, 2014.

4 Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 218–230, 2015.

5 Alfred V. Aho, Daniel S. Hirschberg, and Jeffrey D. Ullman. Bounds on the complexity of the longest common subsequence problem. *Journal of the ACM*, 23(1):1–12, 1976. doi:10.1145/321921.321922.

6 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC'15)*, pages 51–58, 2015. doi:10.1145/2746539.2746612.

7 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Patrascu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014. doi:10.1007/s00453-012-9734-3.

8 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

9 Karl Bringmann. A near-linear pseudopolynomial time algorithm for Subset Sum. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 1073–1084, 2017. doi:10.1137/1.9781611974782.69.

10 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and Dynamic Time Warping. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 79–97, 2015.

11 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the Strong Exponential Time Hypothesis and consequences for non-reducibility. In *Proc. 7th ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 261–270, 2016. doi:10.1145/2840728.2840746.

12 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. 47th Annual ACM Symposium on Theory of Computing, (STOC'15)*, pages 31–40, 2015. doi:10.1145/2746539.2746568.

13 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min,+)-convolution. *ArXiv e-prints*, February 2017. arXiv:1702.07669.

14 Mark de Berg, Kevin Buchin, Bart M. P. Jansen, and Gerhard J. Woeginger. Fine-grained complexity analysis of two classic TSP variants. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, pages 5:1–5:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.5.

15 David Eppstein. Sequence comparison with mixed convex and concave costs. *J. Algorithms*, 11(1):85–101, 1990. doi:10.1016/0196-6774(90)90031-9.

16 David A. Eppstein. *Efficient algorithms for sequence analysis with concave and convex gap costs*. PhD thesis, Columbia University, 1989.

17 Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975. doi:10.1016/0012-365X(75)90103-X.

18 Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

19 Zvi Galil and Raffaele Giancarlo. Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science*, 64(1):107–118, 1989. doi:10.1016/0304-3975(89)90101-1.

20 Zvi Galil and Kunsoo Park. A linear-time algorithm for concave one-dimensional dynamic programming. *Inf. Process. Lett.*, 33(6):309–311, 1990. doi:10.1016/0020-0190(90)90215-J.

21 Zvi Galil and Kunsoo Park. Parallel algorithms for dynamic programming recurrences with more than $O(1)$ dependency. *J. Parallel Distrib. Comput.*, 21(2):213–222, 1994. doi:10.1006/jpdc.1994.1053.

22 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 2162–2181, 2017. doi:10.1137/1.9781611974782.141.

23 Allan Grønlund, Kasper Green Larsen, Alexander Mathiasen, Jesper Sindahl Nielsen, Stefan Schneider, and Mingzhou Song. Fast Exact k-Means, k-Medians and Bregman Divergence Clustering in 1D. *ArXiv e-prints*, January 2017. arXiv:1701.07204.

24 Daniel S. Hirschberg and Lawrence L. Larmore. The least weight subsequence problem. *SIAM Journal on Computing*, 16(4):628–638, 1987. doi:10.1137/0216043.

25 Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. 0-1 Integer Linear Programming with a linear number of constraints. *ArXiv e-prints*, January 2014. arXiv:1401.5512.

26 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

27 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

28 Alon Itai, Christos H Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

29 Maria M. Klawe and Daniel J. Kleitman. An almost linear time algorithm for generalized matrix searching. *SIAM J. Discrete Math.*, 3(1):81–97, 1990. doi:10.1137/0403009.

30 Donald E. Knuth and Michael F. Plass. Breaking paragraphs into lines. *Softw., Pract. Exper.*, 11(11):1119–1184, 1981. doi:10.1002/spe.4380111102.

31 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-grained Complexity of One-Dimensional Dynamic Programming. *ArXiv e-prints*, March 2017. arXiv:1703.00941.

32 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

33 William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980. doi:10.1016/0022-0000(80)90002-1.

34 Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(1):315–334, 1992.

35 Webb Miller and Eugene W. Myers. Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology*, 50(2):97–120, 1988. doi:10.1007/BF02459948.

36 David Pisinger. Dynamic programming on the word RAM. *Algorithmica*, 35(2):128–145, 2003. doi:10.1007/s00453-002-0989-y.

37 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the Strong Exponential Time Hypothesis (invited talk). In *Proc. 10th International Symposium on Parameterized and Exact Computation (IPEC'15)*, pages 17–29, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

38 Robert E. Wilber. The concave least-weight subsequence problem revisited. *J. Algorithms*, 9(3):418–425, 1988. doi:10.1016/0196-6774(88)90032-6.

39 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

40 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 645–654, 2010. doi:10.1109/FOCS.2010.67.

41 F. Frances Yao. Efficient dynamic programming using quadrangle inequalities. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC'80)*, pages 429–435, 1980. doi:10.1145/800141.804691.

On Problems Equivalent to (min,+)-Convolution*

Marek Cygan[1], Marcin Mucha[2], Karol Węgrzycki[3], and Michał Włodarczyk[4]

1 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
cygan@mimuw.edu.pl
2 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
mucha@mimuw.edu.pl
3 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
k.wegrzycki@mimuw.edu.pl
4 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
m.wlodarczyk@mimuw.edu.pl

Abstract

In the recent years, significant progress has been made in explaining apparent hardness of improving over naive solutions for many fundamental polynomially solvable problems. This came in the form of conditional lower bounds – reductions from a problem assumed to be hard. These include 3SUM, All-Pairs Shortest Paths, SAT and Orthogonal Vectors, and others.

In the $(\min,+)$-convolution problem, the goal is to compute a sequence $(c[i])_{i=0}^{n-1}$, where $c[k] = \min_{i=0,\ldots,k}\{a[i] + b[k-i]\}$, given sequences $(a[i])_{i=0}^{n-1}$ and $(b[i])_{i=0}^{n-1}$. This can easily be done in $\mathcal{O}(n^2)$ time, but no $\mathcal{O}(n^{2-\varepsilon})$ algorithm is known for $\varepsilon > 0$. In this paper we undertake a systematic study of the $(\min,+)$-convolution problem as a hardness assumption.

As the first step, we establish equivalence of this problem to a group of other problems, including variants of the classic knapsack problem and problems related to subadditive sequences. The $(\min,+)$-convolution has been used as a building block in algorithms for many problems, notably problems in stringology. It has also already appeared as an ad hoc hardness assumption. We investigate some of these connections and provide new reductions and other results.

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

Keywords and phrases fine-grained complexity, knapsack, conditional lower bounds, (min,+)-convolution, subquadratic equivalence

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.22

1 Introduction

1.1 Hardness in P

For many problems there exist ingenious algorithms that significantly improve upon the naive approach in terms of time complexity. On the other hand, for some fundamental problems, the naive algorithms are still the best known, or have been improved upon only slightly. To some extent this has been explained by the P≠NP conjecture. However, for many problems even the naive approaches lead to polynomial algorithms, and the P≠NP conjecture does not seem to be particularly useful for proving polynomial lower bounds.

* This work is part of a project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 677651).

In the recent years, significant progress has been made in establishing such bounds, conditioned on conjectures other than P≠NP, each of them claiming time complexity lower bounds for a different problem. And so, conjecture that there is no $\mathcal{O}(n^{2-\epsilon})$ algorithm for 3SUM problem[1] implies hardness for problems in computational geometry [22] and dynamic algorithms [33]. The conjecture that All-Pairs Shortest Paths (APSP) is hard implies hardness of finding graph radius, graph median and some dynamic problems (see [38] for survey). Finally, the Strong Exponential Time Hypothesis (SETH) introduced in [25, 26] that has been used extensively to prove hardness of parametrized problems, recently lead to polynomial lower bounds via the intermediate Orthogonal Vectors problem (see [36]). These include bounds for Edit Distance [3], Longest Common Subsequence [9, 2], and other [38].

It is worth noting that in many cases the results mentioned are not only showing the hardness of the problem in question, but also that it is computationally equivalent to the underlying hard problem. This leads to clusters of equivalent problems being formed, each cluster corresponding to a single hardness assumption (see [38, Figure 1]).

As Christos H. Papadimitriou is quoted to say „*There is nothing wrong with trying to prove that P=NP by developing a polynomial-time algorithm for an NP-complete problem. The point is that without an NP-completeness proof we would be trying the same thing without knowing it!*" [32]. In the same spirit, these new conditional hardness results have cleared the polynomial landscape by showing that there really are not that many hard problems.

1.2 Hardness of MinConv

In this paper we propose yet another hardness assumption in the MinConv problem.

MinConv
Input: Sequences $(a[i])_{i=0}^{n-1}$, $(b[i])_{i=0}^{n-1}$
Task: Output sequence $(c[i])_{i=0}^{n-1}$, such that $c[k] = \min_{i+j=k}(a[i] + b[j])$

This problem has been used as a hardness assumption before for at least two specific problems [29, 4], but to the best of our knowledge no attempts have been made to systematically study the neighborhood of this problem in the polynomial complexity landscape. To be more precise, we consider the following.

▶ **Conjecture 1.** *There is no $\mathcal{O}(n^{2-\varepsilon})$ algorithm for* MinConv*, for $\varepsilon > 0$.*

Let us first look at the place occupied by MinConv in the landscape of established hardness conjectures. Figure 1 shows known reductions between these conjectures and includes MinConv. Bremner et al. [7] showed reduction from MinConv to APSP. It is also known [4, 1] that MinConv can be reduced to 3SUM (to the best of our knowledge no such reduction has been published before, and we provide the details in the full version of this paper [18]). Note that a reduction from 3SUM or APSP to MinConv would imply a reduction between 3SUM and APSP, which is a major open problem in the area [38]. No relation is known between MinConv and SETH or OV.

In this paper we study three broad categories of problems. The first category consists of the classic 0/1 Knapsack and its variants, which we show to be essentially equivalent to MinConv. This is perhaps somewhat surprising, given recent progress of Bringmann [8] for SubsetSum, which is a special case of 0/1 Knapsack. However, note that Bringmann's

[1] We included all problem definitions together with known results concerning these problems in Section 2. This is to keep the introduction relatively free of technicalities.

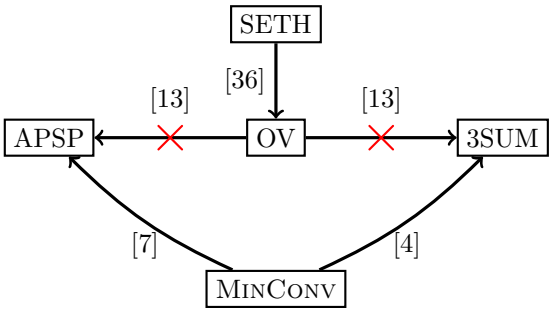


Figure 1 The relationship between popular conjectures. A reduction from OV to 3SUM or APSP contradicts the nondeterministic version of SETH [13, 38] (these arrows are striked-out).

algorithm [8] (as well as in other efficient solutions for SubsetSum) is built upon the idea of composing solutions using the $(\vee, \wedge)$-convolution, which can implemented efficiently using Fast Fourier Transform (FFT). The corresponding composition operation for 0/1 Knapsack is MinConv (see the full version of this paper for details [18]).

The second category consists of problems directly related to MinConv. This includes decision versions of MinConv, and problems related to the notion of subadditivity. Any subadditive sequence a with $a[0] = 0$ is an idempotent of MinConv, so it is perhaps natural that these problems turn out to be equivalent to MinConv.

Finally, we investigate problems that have previously been shown to be related to MinConv, and contribute some new reductions, or simplify existing ones.

2 Problem definitions and known results

2.1 3SUM

3sum
Input: Sets of integers A, B, C, each of size n
Task: Decide whether there exist $a \in A$, $b \in B$, $c \in C$ such that $a + b = c$

The 3sum problem is the first problem that was considered as a hardness assumption in P. It admits a simple $\mathcal{O}(n^2 \log n)$ algorithm but the existence of an $\mathcal{O}(n^{2-\epsilon})$ algorithm remains a big open problem. The first lower bounds based on hardness of 3sum appeared in 1995 [22] and some other examples can be found in [5, 33, 39]. The current best algorithm for 3sum runs in slightly subquadratic expected time $\mathcal{O}\left((n^2/\log^2 n)(\log\log n)^2\right)$ [5]. An $\mathcal{O}\left(n^{1.5}\text{polylog}(n)\right)$ algorithm is possible on the nondeterministic[2] Turing machine [13]. The 3sum problem is known to be subquadratically equivalent to its convolution version in the randomized setting [33].

3sumConv
Input: Sequences a, b, c, each of length n
Task: Decide whether there exist i, j such that $a[i] + b[j] = c[i + j]$

Both problems are sometimes considered with real weights but in this work we restrict only to the integer setting.

[2] We say that decision problem L admits a nondeterministic algorithm in time $T(n)$ if $L \in \text{NTIME}(T(n)) \cap \text{co-NTIME}(T(n))$.

2.2 MinConv

We have already defined the MinConv problem in Subsection 1.2. Note that it is equivalent (just by negating elements) to the analogous MaxConv problem.

MaxConv
Input: Sequences $(a[i])_{i=0}^{n-1}$, $(b[i])_{i=0}^{n-1}$
Task: Output sequence $(c[i])_{i=0}^{n-1}$, such that $c[k] = \max_{i+j=k}(a[i] + b[j])$

We describe our contribution in terms of MinConv as this version has been already been heavily studied. However, in the theorems and proofs we use MaxConv, as it is easier to work with. We will also work with a decision version of the problem.

MaxConv UpperBound
Input: Sequences $(a[i])_{i=0}^{n-1}$, $(b[i])_{i=0}^{n-1}$, $(c[i])_{i=0}^{n-1}$
Task: Decide whether $c[k] \geq \max_{i+j=k}(a[i] + b[j])$ for all k

If we replace the latter condition with $c[k] \leq \max_{i+j=k}(a[i]+b[j])$ we obtain a similar problem MaxConv LowerBound. Yet another statement of a decision version asks whether a given sequence is a self upper bound with respect to MaxConv, i.e., if it is superadditive. From the perspective of MinConv we may ask an analogous question about being subadditive (again equivalent by negating elements). As far as we know, the computational complexity of these problems has not been studied yet.

SuperAdditivity Testing
Input: A sequence $(a[i])_{i=0}^{n-1}$
Task: Decide whether $a[k] \geq \max_{i+j=k}(a[i] + a[j])$ for all k

In the standard $(+, \cdot)$ ring, convolution can be computed in $\mathcal{O}(n \log n)$ time by the FFT. A natural line of attacking MinConv would be to design an analogue of FFT in the $(\min, +)$-semiring, also called a *tropical semiring*[3]. However, due to the lack of inverse for the min-operation it is unclear if such a transform exists for general sequences. When restricted to convex sequences, one can use a tropical analogue of FFT, namely the Legendre-Fenchel transform [19], which can be performed in linear time [30]. Also, [24] considered sparse variants of convolutions and connection with 3sum.

There has been a long line of research dedicated to improve $\mathcal{O}(n^2)$ algorithm for MinConv. Bremner et al. [7] gave an $\mathcal{O}(n^2/\log n)$ algorithm for MinConv, and gave a reduction from MinConv to APSP [7, Theorem 13]. Williams [37] gave an $\mathcal{O}(n^3/2^{\Omega(\log n)^{1/2}})$ algorithm for APSP, which implies the best known $\mathcal{O}(n^2/2^{\Omega(\log n)^{1/2}})$ algorithm for MinConv [15].

Truly subquadratic algorithms for MinConv exist for monotone increasing sequences with integer values bounded by $\mathcal{O}(n)$. Chan and Lewenstein [15] presented an $\mathcal{O}(n^{1.859})$ randomized algorithm and an $\mathcal{O}(n^{1.864})$ deterministic algorithm for that case. They exploited ideas from additive combinatorics. Bussieck et al. [12] showed that for random input, MinConv can be computed in $\mathcal{O}(n \log n)$ expected and $\Theta(n^2)$ worst case time.

If we are satisfied with computing c with a relative error $(1 + \epsilon)$ then general MinConv admits a nearly-linear algorithm [4, 40]. It could be called an FPTAS (fully polynomial-time

[3] In this setting MinConv is often called $(\min, +)$-convolution, inf-convolution or epigraphic sum.

approximation schema) with a remark that usually this name is reserved for single-output problems for which decision versions are NP-hard.

Using techniques of Carmosino et al. [13] and reduction from MaxConv UpperBound to 3sum one can construct an $\mathcal{O}\left(n^{1.5}\text{polylog}(n)\right)$ algorithm working on nondeterministic Turing machines for MaxConv UpperBound (see the full version of this paper [18]). This running time matches the $\mathcal{O}(n^{1.5})$ algorithm for MinConv in the nonuniform decision tree model [7]. This result is based on the techniques of Fredman [21, 20]. It remains unclear how to transfer these results to the word-RAM model [7].

2.3 Knapsack

0/1 Knapsack
Input: A set of items $\mathcal{I}$ with given weights and values $((w_i, v_i))_{i\in\mathcal{I}}$, capacity t
Task: Find the maximal total value of the items subset $\mathcal{I}' \subseteq \mathcal{I}$ such that $\sum_{i\in\mathcal{I}'} w_i \leq t$

If we are allowed to take multiple copies of a single item then we obtain the Unbounded Knapsack problem. The decision versions of both problems are known to be NP-hard [23] but there are classical algorithms based on dynamic programming with a pseudo-polynomial running time $\mathcal{O}(nt)$ [6]. In fact they solve more general problems, i.e., 0/1 Knapsack$^+$ and Unbounded Knapsack$^+$, where we are asked to output answers for each $0 < t' \leq t$. There is also a long line of research on FPTAS for Knapsack with the current best running times respectively $\mathcal{O}(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^3} \log^2 \frac{1}{\epsilon})$ for 0/1 Knapsack [28] and $\mathcal{O}(n + \frac{1}{\epsilon^2} \log^3 \frac{1}{\epsilon})$ for Unbounded Knapsack [27].

2.4 Other problems related to MinConv

Tree Sparsity
Input: A rooted tree T with a weight function $x : V(T) \to \mathbb{N}_{\geq 0}$, parameter k
Task: Find the maximal total weight of rooted subtree of size k

The Tree Sparsity problem admits an $\mathcal{O}(nk)$ algorithm, which was at first invented for restricted case of balanced trees [14] and generalised later [4]. There is also a nearly-linear FPTAS based on the FPTAS for MinConv [4]. It is known that an $\mathcal{O}(n^{2-\epsilon})$ algorithm for Tree Sparsity entails a subquadratic algorithm for MinConv [4].

MCSP
Input: A sequence $(a[i])_{i=0}^{n-1}$
Task: Output the maximal sum of k consecutive elements for each k

There is a trivial $\mathcal{O}(n^2)$ algorithm for MCSP and a nearly-linear FPTAS based on the FPTAS for MinConv [16]. To the best of our knowledge, this is the first problem to have been explicitly proven to be subquadratically equivalent with MinConv [29]. Our reduction to SuperAdditivity Testing allows us to significantly simplify the proof (see Section 6.1).

l_p-Necklace Alignment
Input: Sequences $(x[i])_{i=0}^{n-1}$, $(y[i])_{i=0}^{n-1}$ describing locations of beads on a circle
Task: Output the cost of the best alignment in p-norm, i.e., $\sum_{i=0}^{n-1} d\left(x[i] + c, y[\pi(i)]\right)^p$ where c is a circular shift, π is a permutation, and d is a distance function on a circle

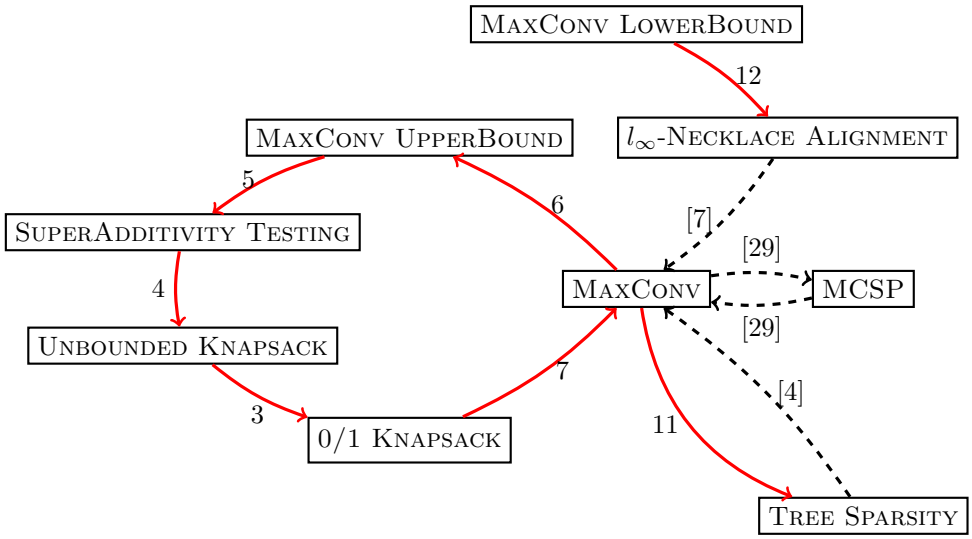


Figure 2 Summary of reductions in the MinConv complexity class. An arrow from problem A to B denotes a reduction from A to B. Black dashed arrows were previously known, red arrows are new results. Numbers next to red arrows point to the corresponding theorems. The only randomized reduction is in the proof of Theorem 7.

For $p = \infty$ we are interested in bounding the maximal distance between any two matched beads. The problem initially emerged for $p = 1$ during the research on geometry of musical rhythm [35]. The family of Necklace Alignment problems has been systematically studied by Bremner et al. [7] for various values of p, in particular $1, 2, \infty$. For $p = 2$ they presented an $\mathcal{O}(n \log n)$ algorithm based on Fast Fourier Transform. For $p = \infty$ the problem was reduced to MinConv which led to a slightly subquadratic algorithm.

Although it is more natural to state the problem with inputs from $[0, 1)$, we find it more convenient to work with integer sequences that describe a necklace after scaling.

Fast $o(n^2)$ algorithms for MinConv have also found applications in text algorithms. Moosa and Rahman [31] reduced the *Indexed Permutation Matching* to MinConv and obtained $o(n^2)$ algorithm. Burcsi et al. [10] used MinConv to get faster algorithms for *Jumbled Pattern Matching* and described how finding dominating pairs can be used to solve MinConv. Later Burcsi et al. [11] showed that fast MinConv can also be used to get faster algorithms for a decision version of the *Approximate Jumbled Pattern Matching* over binary alphabets.

3 New results summary

Figure 2 illustrates the technical contributions of this paper. The long ring of reductions on the left side of the Figure 2 is summarized below.

▶ **Theorem 2.** *The following statements are equivalent:*

1. *There exists an* $\mathcal{O}(n^{2-\varepsilon})$ *algorithm for* MaxConv *for some* $\varepsilon > 0$.
2. *There exists an* $\mathcal{O}(n^{2-\varepsilon})$ *algorithm for* MaxConv UpperBound *for some* $\varepsilon > 0$.
3. *There exists an* $\mathcal{O}(n^{2-\varepsilon})$ *algorithm for* SuperAdditivity Testing *for some* $\varepsilon > 0$.
4. *There exists an* $\mathcal{O}((n+t)^{2-\varepsilon})$ *algorithm for* Unbounded Knapsack *for some* $\varepsilon > 0$.
5. *There exists an* $\mathcal{O}((n+t)^{2-\varepsilon})$ *algorithm for* 0/1 Knapsack *for some* $\varepsilon > 0$.

We allow randomized algorithms.

Theorem 2 is split into five implications, presented separately as Theorems 3,4,5,6 and 7 in Section 5. While Theorem 2 has a relatively short and simple statement, it is not the strongest possible version of the equivalence. In particular, one can show analogous implications for subpolynomial improvements, such as the $\mathcal{O}(n^2/2^{\Omega(\log n)^{1/2}})$ algorithm for MinConv of Williams [37]. The theorems listed above contain stronger versions of the implications.

Section 6 is devoted to the remaining arrows in Figure 2. In Subsection 6.1, we show that by using Theorem 2 we can obtain an alternative proof of the equivalence of MCSP and MaxConv (and so also MinConv), much simpler than the one presented in [29]. In Subsection 6.2, we show that Tree Sparsity reduces to MaxConv, complementing the opposite reduction showed in [4]. Finally in Subsection 6.3 we provide some observations on the possible equivalence between l_∞-Necklace Alignment and MaxConv.

4 Preliminaries

We present a series of results of the following form: if a problem $\mathcal{A}$ admits an algorithm with running time $T(n)$, then a problem $\mathcal{B}$ admits an algorithm with running time $T'(n)$, where function T' depends on T and n is the length of the input. Our main interest is in showing that $T(n) = \mathcal{O}(n^{2-\epsilon}) \Rightarrow T'(n) = \mathcal{O}(n^{2-\epsilon'})$. Some problems, in particular Knapsack, have no simple parameterization and we allow function T to take multiple arguments.

We assume that for all studied problems the input consists of a list of integers within $[-W, W]$. For the sake of readability we omit W as a running time parameter and we allow function T to hide polylog(W) factors. As sometimes the size of the input grows in the reduction, we restrict ourselves to a class of functions satisfying $T(cn) = \mathcal{O}(T(n))$ for a constant c. This is justified as we mainly focus on functions of the form $T(n) = n^\alpha$. In some reductions the integers in the new instance may increase to $\mathcal{O}(nW)$. In that case we multiply the running time by polylog(n) to take into account the overhead of performing arithmetic operations. All logarithms are base 2.

5 Main reductions

▶ **Theorem 3** (Unbounded Knapsack $\to$ 0/1 Knapsack). *A $T(n,t)$ algorithm for* 0/1 Knapsack *implies an $\mathcal{O}(T(n,t)\log t)$ algorithm for* Unbounded Knapsack.

Proof. Consider an instance of Unbounded Knapsack with the capacity t and the set of items given as weight-value pairs $((w_i, v_i))_{i\in\mathcal{I}}$. Construct an equivalent 0/1 Knapsack instance with the same t and the set of items $\left((2^j w_i, 2^j v_i)\right)_{\in\mathcal{I}, 0\le j\le \log t}$. Let $X = (x_i)_{\in\mathcal{I}}$ be the list of multiplicities of items chosen in a solution to the Unbounded Knapsack problem. Of course $x_i \le t$. Define $(x_i^j)_{0\le j\le \log t}$, $x_i^j \in \{0,1\}$ to be the binary representation of x_i. Then the vector $(x_i^j)_{\in\mathcal{I}, 0\le j\le \log t}$ induces a solution to 0/1 Knapsack with the same total weight and value. The described mapping can be reverted what implies the equivalence between the instances and proves the claim. ◀

▶ **Theorem 4** (SuperAdditivity Testing $\to$ Unbounded Knapsack). *If* Unbounded Knapsack *can be solved in time $T(n,t)$ then* SuperAdditivity Testing *admits an algorithm with running time $\mathcal{O}(T(n,n)\log n)$.*

Proof. Let $(a[i])_{i=0}^{n-1}$ be a non-negative monotonic sequence.[4] Set $D = \sum_{i=0}^{n-1} a[i] + 1$ and construct an Unbounded Knapsack instance with the set of items $((i, a[i]))_{i=0}^{n-1} \cup$

[4] For a technical reduction of SuperAdditivity Testing to this case see the full version of this paper [18].

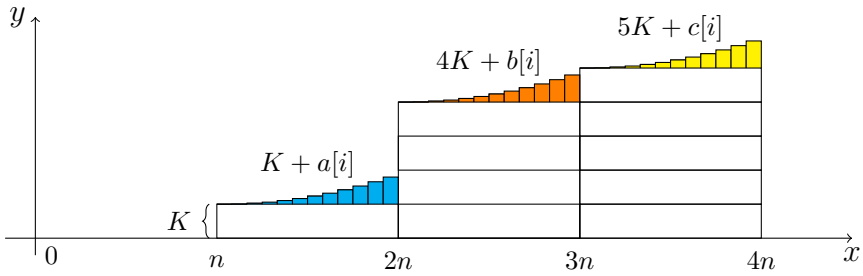


Figure 3 Graphical interpretation of the sequence e in Theorem 5. The height of rectangles equals K.

$((2n-1-i, D-a[i]))_{i=0}^{n-1}$ and $t = 2n-1$. It is always possible to gain D by taking two items $(i, a[i]), (2n-1-i, D-a[i])$ for any i. We will claim that the answer to the constructed instance equals D if and only if a is superadditive.

If a is not superadditive, then there are i, j such that $a[i] + a[j] > a[i+j]$. Choosing $((i, a[i]), (j, a[j]), (2n-1-i-j, D-a[i+j]))$ gives a solution of value exceeding D.

Now assume that a is superadditive. Observe that any feasible knapsack solution may contain at most one item with weight exceeding $n-1$. On the other hand, the optimal solution has to include one such item because the total value of the lighter ones is less than D. Therefore the optimal solution contains an item $(2n-1-k, D-a[k])$ for some $k < n$. The total weight of the rest of the solution is at most k. As a is superadditive, we can replace any pair $(i, a[i]), (j, a[j])$ with the item $(i+j, a[i+j])$ without decreasing the value of the solution. By repeating this argument, we end up with a single item lighter than n. The sequence a is monotonic so it is always profitable to replace this item with a heavier one, as long as the load does not exceed t. We conclude that the optimal solution must be of form $((k, a[k]), (2n-1-k, D-a[k]))$, which completes the proof. ◀

▶ **Theorem 5** (MaxConv UpperBound → SuperAdditivity Testing). *If* SuperAdditivity Testing *can be solved in time* $T(n)$ *then* MaxConv UpperBound *admits an algorithm with running time* $\mathcal{O}(T(n)\log n)$.

Proof. We start with reducing the instance of MaxConv UpperBound to the case of non-negative monotonic sequences. Observe that condition $a[i] + b[j] \le c[i+j]$ can be rewritten as $(C + a[i] + Di) + (C + b[j] + Dj) \le 2C + c[i+j] + D(i+j)$ for any constants C, D. Hence, replacing sequences $(a[i])_{i=0}^{n-1}$, $(b[i])_{i=0}^{n-1}$, $(c[i])_{i=0}^{n-1}$ with $a'[i] = C + a[i] + Di$, $b'[i] = C + b[i] + Di$, $c'[i] = 2C + c[i] + Di$ leads to an equivalent instance. We can thus pick C, D of magnitude $\mathcal{O}(W)$ to ensure that all elements are non-negative and do not exceed the successor. The values in the new sequences may rise up to $\mathcal{O}(nW)$.

From now we can assume the given sequences to be non-negative and monotonic. Define K to be the maximal value occurring in any sequence. Construct a sequence e of length $4n$ as follows. For $i \in [0, n-1]$ set $e[i] = 0$, $e[n+i] = K + a[i]$, $e[2n+i] = 4K + b[i]$, $e[3n+i] = 5K + c[i]$. If there is $a[i] + b[j] > c[i+j]$ for some i, j, then $e[n+i] + e[2n+j] > e[3n+i+j]$ and therefore e is not superadditive. We now show that otherwise e must be superadditive.

Assume w.l.o.g. $i \le j$. The case $i < n$ can be ruled out because it implies $e[i] = 0$ and $e[i] + e[j] \le e[i+j]$ for any j as e is monotonic. If $i \ge 2n$, then $i + j \ge 4n$, so we can restrict to $i \in [n, 2n-1]$. We can also clearly assume $j < 3n$. If $j \in [n, 2n-1]$, then $e[i] + e[j] \le 4K \le e[i+j]$. Finally, $j \in [2n, 3n-1]$ corresponds to the original condition. ◀

▶ **Theorem 6** (MaxConv → MaxConv UpperBound). *A $T(n)$ algorithm for* MaxConv UpperBound *implies an $\mathcal{O}(T(\sqrt{n})n\log n)$ algorithm for* MaxConv.

The proof of the reduction from MaxConv to MaxConv UpperBound has been independently given recently in [4]. For completeness we give our proof in the full version of this paper [18].

▶ **Theorem 7** (0/1 Knapsack → MaxConv). *A $T(n)$ algorithm for* MaxConv *implies an $\mathcal{O}(T(t\log t)\log^3(n/\delta)\log n)$ for* 0/1 Knapsack *that outputs the correct answer with probability at least $1-\delta$.*

▶ **Corollary 8.** *The $\mathcal{O}((n+t)^{2-\epsilon})$ time algorithm for* 0/1 Knapsack *implies the randomized $\mathcal{O}(t^{2-\epsilon'}+n)$ time algorithm for* 0/1 Knapsack$^+$.

The proof follows the approach of Bringmann [8], and we present it in the full version of this paper [18].

6 Other problems related to MinConv

6.1 Maximum consecutive subsums problem

The Maximum Consecutive Subsums Problem (MCSP) is to the best of our knowledge the first problem that has been explicitly proven to be subquadratically equivalent with MinConv [29]. The reduction from MCSP to MaxConv is only shown for completeness, but the reduction in the opposite direction is much simpler than the original one.

▶ **Theorem 9** (MCSP → MaxConv). *If* MaxConv *can be solved in time $T(n)$ then* MCSP *admits an algorithm with running time $\mathcal{O}(T(n))$.*

Proof. Let $(a[i])_{i=0}^{n-1}$ be the input sequence. Construct sequences of length $2n$ as follows: $b[k]=\sum_{i=0}^{k}a[i]$ for $k<n$, $c[k]=-\sum_{i=0}^{n-k-1}a[i]$ for $k\le n$ (empty sum equals 0) and otherwise $b[k]=c[k]=-D$, where D is two times larger than any partial sum. Observe that

$$(b\oplus^{\max}c)[n+k-1]=\max_{\substack{0\le j<n\\ 0\le n+k-j-1\le n}}\sum_{i=0}^{j}a[i]-\sum_{i=0}^{j-k}a[i]=\max_{k-1\le j<n}\sum_{i=j-k+1}^{j}a[i], \tag{1}$$

so we can read the maximum consecutive sum for each length k after performing MaxConv. ◀

▶ **Theorem 10** (SuperAdditivity Testing → MCSP). *If* MCSP *can be solved in time $T(n)$ then* SuperAdditivity Testing *admits an algorithm with running time $\mathcal{O}(T(n))$.*

Proof. Let $(a[i])_{i=0}^{n-1}$ be the input sequence and $b[i]=a[i+1]-a[i]$. The superadditivity condition $a[k]\le a[k+j]-a[j]$ (for all possible k,j) can be translated into $a[k]\le\min_{0\le j<n-k}\sum_{i=j}^{k+j-1}b[i]$ (for all k), so computing MCSP vector on $(-b[i])_{i=0}^{n-2}$ suffices to check if the above condition holds. ◀

6.2 Tree Sparsity

▶ **Theorem 11** (Tree Sparsity → MaxConv). *If* MaxConv *can be solved in time $T(n)$ and the function T is superadditive then* Tree Sparsity *admits an algorithm with running time $\mathcal{O}\left(T(n)\log^2 n\right)$.*

Proof. We take advantage of the heavy-light decomposition introduced by Sleator and Tarjan [34]. This technique has been utilized by Backurs et al. [4] in order to transform a nearly-linear PTAS for MaxConv to a nearly-linear PTAS for Tree Sparsity. The reduction for exact subquadratic algorithms is different in the second phase though.

We construct a *spine* with a *head* s_1 at the root of the tree. We define s_{i+1} to be the child of s_i with the larger subtree (in case of draw we choose any child) and the last node in the spine is a leaf. The remaining children of nodes s_i become heads for analogous spines so the whole tree gets covered. Note that every path from a leaf to the root intersects at most $\log n$ spines because each spine transition doubles the subtree size.

For a node v with a subtree of size m we define the sparsity vector $(x^v[0], x^v[1], \ldots, x^v[m])$ with the weights of the heaviest subtrees rooted at v with fixed sizes. We are going to compute sparsity vectors for all heads of spines in the tree recursively. Let $(s_i)_{i=1}^{\ell}$ be a spine with a head v and let u^i indicate the sparsity vector for the child of s_i being a head (i.e., the child with the smaller subtree). If s_i has less than two children we treat u^i as a vector (0).

For an interval $[a, b] \subseteq [1, \ell]$ let $u^{a,b} = u^a \oplus^{\max} u^{a+1} \oplus^{\max} \cdots \oplus^{\max} u^b$ and $y^{a,b}[k]$ be the maximum weight of a subtree of size k rooted at s_a and not containing s_{b+1}. Let $c = \left\lfloor \frac{a+b}{2} \right\rfloor$. The $\oplus^{\max}$ operator is associative so $u^{a,b} = u^{a,c} \oplus^{\max} u^{c+1,b}$. To compute the second vector we consider two cases: whether the optimal subtree contains s_{c+1} or not.

$$\begin{aligned} y^{a,b}[k] &= \max\left[y^{a,c}[k], \quad \sum_{i=a}^{c} x(s_i) + \max_{k_1+k_2=k-(c-a+1)} \left(u^{a,c}[k_1] + y^{c+1,b}[k_2] \right) \right] \\ &= \max\left[y^{a,c}[k], \quad \sum_{i=a}^{c} x(s_i) + \left(u^{a,c} \oplus^{\max} y^{c+1,b} \right)[k-(c-a+1)] \right] \end{aligned}$$

Using the presented formulas we reduce the problem of computing $x^v = y^{1,\ell}$ to subproblems for intervals $[1, \frac{\ell}{2}]$ and $[\frac{\ell}{2}+1, \ell]$ and results are merged with two $(\max, +)$-convolutions. Proceeding further we obtain $\log \ell$ levels of recursion, where the sum of convolution sizes on each level is $\mathcal{O}(m)$, what results in the total running time $\mathcal{O}(T(m) \log m)$ (recall that T is superadditive).

The second type of recursion comes from the spine decomposition. There are at most $\log n$ levels of recursion with the cumulative sum of subtrees bounded by n on each level, what proves the claim. ◀

6.3 l_∞-Necklace Alignment

In this section we study the l_∞-Necklace Alignment alignment problem that was proved to reduce to MinConv [7]. We are unable to reduce any of the problems equivalent to MinConv to this problem, but we do reduce a related problem - MaxConv LowerBound. We also elaborate on why obtaining a full reduction is difficult.

▶ **Theorem 12** (MaxConv LowerBound $\to$ l_∞-Necklace Alignment)**.** *If* l_∞-Necklace Alignment *can be solved in time* $T(n)$ *then* MaxConv LowerBound *admits an algorithm with running time* $\mathcal{O}(T(n) \log n)$.

Proof. Let a, b, c be the input sequences to MaxConv LowerBound. We call a sum of form $e_1[k_1] + e_2[k_2] + \cdots + e_m[k_m]$, where $e_i \in \{a, b, c\}$, a *combination*, and we define its order as $\sum_{i=1}^{m} k_i$. If an element $e_i[k_i]$ occurs with minus, we subtract k_i.

We can assume the following properties of the input sequences w.l.o.g.

1. *We may assume the sequences are non-negative and* $a[i] \le c[i]$ *for all* i. Just add C_1 to a, $C_1 + C_2$ to b, and $2C_1 + C_2$ to c for appropriate positive constants C_1, C_2.

2. *We can artificially append an element $b[n]$ larger than value of any combination of order n and length bounded by a constant L.* Alternatively, we can say that combinations of order 0 with a positive coefficient at $b[n]$ have positive value. Initially, we enforce this property by setting $b[n]$ as the maximum absolute value of an element times L.
3. *Any combination of positive order and length bounded by L has a non-negative value.* Add a linear function Di to all sequences. As the order of combination is positive, the factors at D sum up to a positive value. It suffices to choose D equal to the maximum absolute value of an element times L. Note that previous inequalities compare combinations of the same order so they stay unaffected.

The values of the elements might increase to $\mathcal{O}(nWL^2)$. For the rest of this proof we will use $L = 10$. Let $B = b[n]$, $B_1 = b[n-1]$, $B_2 = b[n] - b[1]$. We define necklaces x, y of length $2B$ with $2n$ beads each. The property (3) implies monotonicity of the sequences so the beads are given in the right order. We allow two beads to lie in the same place (in particular the first one and the last one in y).

$$\begin{array}{lccccccccc} x = (& a[0], & a[1], & \dots, & a[n-1], & B + c[0], & B + c[1], & \dots, & B + c[n-2], & B + c[n-1] \ ), \\ y = (& B_1 - b[n-1], & B_1 - b[n-2], & \dots, & B_1 - b[0], & B + B_2 - b[n-1], & B + B_2 - b[n-2], & \dots, & B + B_2 - b[1], & 2B \ ). \end{array}$$

Bremner et al. [7] pointed out that the optimal solution for l_∞-Necklace Alignment must be non-crossing, so we can consider only matchings of form $(x[i], y[j])$ where $j = i + k \bmod 2n$ and k is fixed. Let $d(x[i], y[j])$ be the forward distance between $x[i]$ and $y[j]$, i.e., $y[j] - x[i]$ plus the length of the necklaces if $j < i$. Define M_k to be $\max_{i\in[0,N)} d\big(x[i], y[k + i \bmod 2n]\big) - \min_{i\in[0,N)} d\big(x[i], y[k + i \bmod 2n]\big)$. In this setting [7, Fact 5] says that for a fixed k the optimal shift provides solution of value $\frac{M_k}{2}$.

We want to show that for $k \in [0, n)$ it holds

$$\begin{aligned} \min_{i\in[0,2n)} d\big(x[i], y[k + i \bmod 2n]\big) &= B_1 - \max_{i+j=n-k-1} (a[i] + b[j]), \\ \max_{i\in[0,2n)} d\big(x[i], y[k + i \bmod 2n]\big) &= B - c[n-k-1]. \end{aligned}$$

There are five types of connections between beads.

$$d\big(x[i], y[k + i \bmod 2n]\big) = \begin{cases} B_1 - a[i] - b[n-k-1-i] & i \in [0, n-k-1], \quad \text{(I)} \\ B + B_2 - a[i] - b[2n-k-1-i] & i \in [n-k, n-1], \quad \text{(II)} \\ B_2 - b[2n-k-1-i] - c[i-n] & i \in [n, 2n-k-2], \quad \text{(III)} \\ B - c[n-k-1] & i = 2n-k-1, \quad \text{(IV)} \\ B + B_1 - b[3n-k-1-i] - c[i-n] & i \in [2n-k, 2n-1]. \quad \text{(V)} \end{cases}$$

All formulas form combinations of length bounded by 5 so we can apply the properties (2,3). Observe that the order of each combination equals k, except for $i = 2n-k-1$ where the order is $k+1$. Using the property (3) we reason that $B - c[n-k-1]$ is indeed the maximal forward distance. It remains to show that the minimum lies within the group (I). Note that these are the only combinations that lack $b[n]$. By the property (2) each distance from the group (I) compares less with any other distance because the combinations have the same order (except for the maximal one) and only the latter contains $b[n]$.

For $k < n$ the condition $M_k < B - B_1$ is equivalent to $c[n-k-1] > \max_{i+j=n-k-1}(a[i] + b[j])$. If there is such a k, i.e., the answer to MaxConv LowerBound for sequences a, b, c is NO, then $\min_k M_k < B - B_1$ and the return value is less than $\frac{1}{2}(B - B_1)$.

Finally, we need to prove that otherwise $M_k \geq B - B_1$ for all k. We have already acknowledged that for $k < n$. Each matching for $k \geq n$ can be represented as swapping sequences a and c inside the necklace x, composed with the index shift by $k - n$. The two halves of the necklace x are analogous so all the prior observations on the matching structure remain valid.

If the answer to MaxConv LowerBound for sequences a, b, c is YES, then $\forall_{k\in[0,n)}\exists_{i+j=k} a[i]+b[j] \geq c[k]$. The property (1) guarantees that $a \leq c$ so we conclude that $\forall_{k\in[0,n)}\exists_{i+j=k} c[i]+b[j] \geq a[i]+b[j] \geq c[k] \geq a[k]$, and by the same argument as before the cost of the solution is at least $\frac{1}{2}(B-B_1)$. ◀

Observe that both l_∞-Necklace Alignment and MaxConv LowerBound admit simple linear nondeterministic algorithms. For MaxConv LowerBound it is enough to either assign each k a single condition $a[i]+b[k-i] \geq c[k]$ that is satisfied, or guess a k for which none inequality holds. For l_∞-Necklace Alignment we define a decision version of the problem by asking if there is an alignment of value bounded by K (the problem is self-reducible via binary search). For positive instances the algorithm just guesses k inducing an optimal solution. For negative instances it must hold $M_k > 2K$ for all k. Therefore, it suffices to guess for each k a pair i, j such that $d\big(x[i], y[k+i \bmod n]\big) - d\big(x[j], y[k+j \bmod n]\big) > 2K$.

We remind that MaxConv UpperBound reduces to 3sum which admits an $\mathcal{O}\left(n^{1.5}\text{polylog}(n)\right)$ nondeterministic algorithm [13] so in fact there is no obstacle for a subquadratic reduction from MaxConv LowerBound to MaxConv UpperBound to exist (see the full version of this paper [18]). However, the nondeterministic algorithm for 3sum exploits techniques significantly different from ours, including modular arithmetic, and a potential reduction would probably need to rely on some different structural properties of MaxConv.

7 Conclusions and future work

In this paper we undertake a systematic study of MinConv as a hardness assumption, and prove subquadratic equivalence of MinConv with SuperAdditivity Testing, Unbounded Knapsack, 0/1 Knapsack, and Tree Sparsity. An intriguing open problem is to establish the relation between the MinConv conjecture and SETH.

One consequence of our results is a new lower bound on 0/1 Knapsack. It is known that an $\mathcal{O}(t^{1-\epsilon}n^{\mathcal{O}(1)})$ algorithm for 0/1 Knapsack contradicts the SetCover conjecture [17]. Here, we show that an $\mathcal{O}((n+t)^{2-\epsilon})$ algorithm contradicts the MinConv conjecture. This does not rule out an $\mathcal{O}(t+n^{\mathcal{O}(1)})$ algorithm, which leads to another interesting open problem.

Finally, it is open whether MaxConv LowerBound is equivalent to MinConv, which would imply an equivalence between l_∞-Necklace Alignment and MinConv.

Acknowledgements. We would like to thank Amir Abboud, Karl Bringmann and Virginia Vassilevska Williams for helpful discussions.

References

1 Amir Abboud. Personal communication.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.14.

3 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58. ACM, 2015. doi:10.1145/2746539.2746612.

4 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better approximations for tree sparsity in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2215–2229. SIAM, 2017.

5 Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic Algorithms for 3SUM. In *Proceedings of the 9th International Conference on Algorithms and Data Structures*, WADS'05, pages 409–421, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/11534273_36.

6 Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

7 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, and Perouz Taslakian. Necklaces, Convolutions, and X + Y. In Yossi Azar and Thomas Erlebach, editors, *Algorithms – ESA 2006: 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006. Proceedings*, pages 160–171, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

8 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1073–1084. SIAM, 2017.

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

10 Peter Burcsi, Ferdinando Cicalese, Gabriele Fici, and Zsuzsanna Lipták. On table arrangements, scrabble freaks, and jumbled pattern matching. In Paolo Boldi and Luisa Gargano, editors, *Fun with Algorithms, 5th International Conference, FUN 2010, Ischia, Italy, June 2-4, 2010. Proceedings*, volume 6099 of *Lecture Notes in Computer Science*, pages 89–101. Springer, 2010. doi:10.1007/978-3-642-13122-6_11.

11 Peter Burcsi, Ferdinando Cicalese, Gabriele Fici, and Zsuzsanna Lipták. On approximate jumbled pattern matching in strings. *Theory Comput. Syst.*, 50(1):35–51, 2012. doi:10.1007/s00224-011-9344-5.

12 Michael R. Bussieck, Hannes Hassler, Gerhard J. Woeginger, and Uwe T. Zimmermann. Fast algorithms for the maximum convolution problem. *Oper. Res. Lett.*, 15(3):133–141, 1994. doi:10.1016/0167-6377(94)90048-5.

13 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270. ACM, 2016. doi:10.1145/2840728.2840746.

14 Coralia Cartis and Andrew Thompson. An exact tree projection algorithm for wavelets. *IEEE Signal Processing Letters*, 20(11):1026–1029, 2013.

15 Timothy M. Chan and Moshe Lewenstein. Clustered Integer 3SUM via Additive Combinatorics. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC'15, pages 31–40, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746568.

16 Ferdinando Cicalese, Eduardo Sany Laber, Oren Weimann, and Raphael Yuster. Approximating the maximum consecutive subsums of a sequence. *Theor. Comput. Sci.*, 525:130–137, 2014. doi:10.1016/j.tcs.2013.05.032.

17 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dańiel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlstrom. On problems as hard as cnf-sat. In *Proceedings of the 2012 IEEE Conference on Computational Complex-*

ity (CCC), CCC'12, pages 74–84, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/CCC.2012.36.

18 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min, +)-convolution. *CoRR*, abs/1702.07669, 2017. URL: http://arxiv.org/abs/1702.07669.

19 Werner Fenchel. On conjugate convex functions. *Canad. J. Math*, 1(73-77), 1949.

20 Michael L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci.*, 1(4):355–361, 1976. doi:10.1016/0304-3975(76)90078-5.

21 Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976. doi:10.1137/0205006.

22 Anka Gajentaan and Mark H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

23 M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. San Francisco: Freeman, 1979.

24 Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 45:1–45:16. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.45.

25 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

27 Klaus Jansen and Stefan E. J. Kraft. A Faster FPTAS for the Unbounded Knapsack Problem. In Zsuzsanna Lipták and William F. Smyth, editors, *Combinatorial Algorithms: 26th International Workshop, IWOCA 2015, Verona, Italy, October 5-7, 2015, Revised Selected Papers*, pages 274–286, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-29516-9_23.

28 Hans Kellerer and Ulrich Pferschy. Improved Dynamic Programming in Connection with an FPTAS for the Knapsack Problem. *Journal of Combinatorial Optimization*, 8(1):5–11, 2004. doi:10.1023/B:JOCO.0000021934.29833.6b.

29 Eduardo Sany Laber, Wilfredo Bardales Roncalla, and Ferdinando Cicalese. On lower bounds for the maximum consecutive subsums problem and the (min, +)-convolution. In *2014 IEEE International Symposium on Information Theory, Honolulu, HI, USA, June 29 – July 4, 2014*, pages 1807–1811. IEEE, 2014. doi:10.1109/ISIT.2014.6875145.

30 Yves Lucet. Faster than the Fast Legendre Transform, the Linear-time Legendre Transform. *Numerical Algorithms*, 16(2):171–185, 1997. doi:10.1023/A:1019191114493.

31 Tanaeem M. Moosa and M. Sohel Rahman. Indexing permutations for binary strings. *Inf. Process. Lett.*, 110(18-19):795–798, 2010. doi:10.1016/j.ipl.2010.06.012.

32 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

33 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

34 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, June 1983. doi:10.1016/0022-0000(83)90006-5.

35 Godfried Toussaint. The geometry of musical rhythm. In Jin Akiyama, Mikio Kano, and Xuehou Tan, editors, *Discrete and Computational Geometry: Japanese Conference,*

JCDCG 2004, Tokyo, Japan, October 8-11, 2004, Revised Selected Papers, pages 198–212, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11589440_20.

36 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

37 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC'14, pages 664–673, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591811.

38 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 17–29. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

39 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing*, 42(3):831–854, 2013.

40 Uri Zwick. All pairs shortest paths in weighted directed graphs-exact and almost exact algorithms. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 310–319. IEEE, 1998.

On Finding the Jaccard Center*

Marc Bury[1] and Chris Schwiegelshohn[2]

1 **Thyssenkrupp Industrial Solutions AG, Essen, Germany**
`marc.bury@thyssenkrupp.com`
2 **Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Rome, Italy**
`chris.schwiegelshohn@tu-dortmund.de`

Abstract

We initiate the study of finding the Jaccard center of a given collection N of sets. For two sets X, Y, the Jaccard index is defined as $|X \cap Y|/|X \cup Y|$ and the corresponding distance is $1 - |X \cap Y|/|X \cup Y|$. The Jaccard center is a set C minimizing the maximum distance to any set of N.

We show that the problem is NP-hard to solve exactly, and that it admits a PTAS while no FPTAS can exist unless $P = NP$. Furthermore, we show that the problem is fixed parameter tractable in the maximum Hamming norm between Jaccard center and any input set. Our algorithms are based on a compression technique similar in spirit to coresets for the Euclidean 1-center problem.

In addition, we also show that, contrary to the previously studied median problem by Chierichetti et al. (SODA 2010), the continuous version of the Jaccard center problem admits a simple polynomial time algorithm.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures

Keywords and phrases Clustering, 1-Center, Jaccard

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.23

1 Introduction

The Jaccard index is a widely used similarity measure on item sets. Given two sets X and Y over a base set U, the similarity is defined as $J(X, Y) = |X \cap Y|/|X \cup Y|$ and the distance is $D(X, Y) = 1 - J(X, Y) = |X \triangle Y|/|X \cup Y|$, where $X \triangle Y$ denotes the symmetric difference of X and Y. In this paper we study the problem of finding the center of a given set of item sets under the Jaccard distance, i.e. for a given collection of sets $N = \{X_1, \ldots, X_n\}$ finding a set $C \subset U$ such that $\max_{X \in N} D(X, C)$ is minimized.

The Jaccard index is arguably the oldest [25] and best known similarity measure on binary data. It has found a wide range of applications such as plagiarism detection [7], association rule mining [12], collaborative filtering [13], web compression [9], biogeographical analysis [34], and chemical similarity searching [39]. Most theoretical computer science research dealing with the Jaccard index focuses on hashing algorithms for nearest neighbor problems, which was pioneered by Broder [6], though a number of publications also deal with clustering tasks on the Jaccard metric, see, for instance, Guha et al [22]. Previous research

* This work was supported by the German Research Council (DFG) within the Collaborative Research Center SFB 876, project A2, and the Google Focused Award on Web Algorithmics for Large-scale Data Analysis.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 23; pp. 23:1–23:14

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

most closely related to this paper addresses the Jaccard median problem, i.e. finding an item set that minimizes the sum of Jaccard distances, see [35, 37].Only recently did Chierichetti et al. [10] show that the Jaccard median problem is NP-hard but also admits a PTAS.

From a more general perspective, the task of finding a single center in a metric space has been studied in various forms dating back to the 19th century [36]. In constant Euclidean space, linear time algorithms exist [30, 38]. In higher dimensions, approximate algorithms based on (weak) coresets have been proposed [3, 4, 11, 26, 40]. Hardness for the 1-center problem in certain finite metrics have been established, including permutation metrics such as Kendall tau and Cayley distances [2, 5, 33], the edit distance on strings [14, 32], and the Hamming metric on strings [16, 27]. The latter problem, also known as the *closest string problem*, is one of the most widely studied center problems in computer science with numerous results on fixed parameter algorithms [15, 21, 29] and approximation algorithms [18, 27, 28]. The more general k-center problem admits a tight 2-approximation in any metric space [19, 23], though some improvements are possible in restricted metrics such as Euclidean space [4].

Our Contribution

We show that the problem is NP-hard to solve exactly, even when the input item sets have cardinality 2. Since the Jaccard distance is a metric, any input point is a trivial 2-approximate solution, and it is easy to see that this bound is tight. We propose two algorithms for the problem. The first algorithm is a PTAS with running time $|N|^{O(\varepsilon^{-6})}|U|^2$. The second one is an FPT algorithm with parameter $k = \max_{X \in N} |X \triangle C|$, i.e. the maximum Hamming norm of input points and Jaccard center C and running time $2^{O(k^3)} \cdot |N| \cdot |U|^3$. As a consequence of our hardness result, we show that under the exponential time hypothesis [24] no FPT algorithm with parameter k and running time $2^{o(k)}$ and no PTAS with running time $2^{o(\sqrt{1/\varepsilon})}$ can exist.

Lastly, we also briefly remark on the continuous version of the problem. Here the input points are non-negative d-dimensional real vectors and $J_c(X,Y) = \frac{\sum_{i=1}^{d} \min(X_i,Y_i)}{\sum_{i=1}^{d} \max(X_i,Y_i)}$. While the Jaccard median problem remains NP-hard for the continuous setting [10, 35], the center problem becomes solvable in polynomial time.

Our Techniques

Our algorithms are based on the existence of a small subset of input points we call *core-covers*. Informally, the union of items of all sets in the *core-cover* contains the (majority of) items of some optimal center C. Specifically, the intersection of the union of items with C yields an α-approximate solution. An *anchored core-cover* further restricts the possible solutions by always containing the items in the intersection of all sets of the core-cover. Crucially, we show that the size of an appropriate (anchored) core-cover is independent of the input when aiming for a $(1+\varepsilon)$-approximation, and dependent only on the parameter k in the context of the FPT algorithm.

In an approximate variant, a core-cover is similar to but weaker than a weak coreset for the Euclidean minimum enclosing ball problem, which requires that the expansion of the minimum enclosing ball computed on the coreset by an $(1+\varepsilon)$-factor contains the entire point set. The existence of constant size weak coresets has been widely studied and utilized [3, 4, 11, 26, 40]. Though the Jaccard distance can be isometrically embedded into (high dimensional) squared Euclidean space, see Gower and Legendre [20], the weak coreset results do not seem to be applicable to the constrained set of solutions corresponding to

embedded item sets. Stronger coreset guarantees extending to arbitrary centers require an exponential dependency on the dimension [1] and therefore also do not seem to be feasible for our purposes.

For the PTAS, we proceed as follows. It turns out that a natural LP relaxation can be efficiently rounded for a large fraction of inputs, namely when for all input sets $X \in N$, we have $\mathrm{OPT} \cdot |X| \in \Omega(\log n/\varepsilon^2)$, where OPT denotes the objective value of the optimum center. If the LP cannot be efficiently rounded, the symmetric difference between any two input sets as well as the optimum set is bounded by $O(\log n/\varepsilon^4)$. A QPTAS now immediately follows by choosing an arbitrary set X, iterating over all subsets S of the base set U with $|S| \in O(\log n/\varepsilon^4)$, and determining the best solution among all candidate centers $X \triangle S$. If we choose multiple sets $X_1, \dots, X_m$ then the number of candidate subsets S will be reduced. In fact, if $X_1, \dots, X_m$ is an anchored core-cover then the dependency on the size of the base set $|U|$ can be replaced by some constant depending only on m and ε. Since there exist anchored core-covers of size $O(1/\varepsilon)$, we obtain a polynomial running time for any fixed ε.

For the FPT-algorithm, the main technical difficulties are to show (1) that the size of an appropriate core-cover can be bounded in terms of the parameter k and (2) that we can efficiently construct an anchored core-cover. As was the case for the PTAS, for a given preliminary anchored core-cover M, we can compute an induced optimum via complete enumeration. If the induced optimum has distance at most OPT to all sets $X \in N$, we are done. Otherwise, any set violating this bound can be added to M. The improvement rate of each added set matches the non-constructive bounds used to show the existence of core-covers, ensuring that the algorithm terminates quickly.

2 Preliminaries

Let $U = \{u_1, \dots u_d\}$ be a base set containing d elements and let $N \subset \mathcal{P}(U)$ be a collection of n subsets of U. Denote the symmetric difference of two sets by $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$.

▶ **Definition 1** (Binary Jaccard Measures)**.** Given $X, Y \subseteq U$, the *Jaccard similarity* is defined as

$$J(X,Y) = \begin{cases} \frac{|X \cap Y|}{|X \cup Y|} & \text{if } X \cup Y \neq \emptyset \\ 1 & \text{if } X \cup Y = \emptyset, \end{cases}$$

and the *Jaccard distance* is defined as $D(X,Y) = 1 - J(X,Y)$.

It is convenient to refer to specific elements of a set X by the characteristic vector $X \in \{0,1\}^d$ where $X_i = 1$ if $u_i \in X$ and $X_i = 0$ otherwise. The extension of the Jaccard measure to vectors with non-negative but otherwise arbitrary entries is as follows.

▶ **Definition 2** (Continuous Jaccard Measures)**.** Given two d dimensional vectors X, Y with non-negative real entries, the *continuous Jaccard similarity* is defined as

$$J_c(X,Y) = \begin{cases} \frac{\sum_{i=1}^d \min(X_i,Y_i)}{\sum_{i=1}^d \max(X_i,Y_i)} & \text{if } \sum_{i=1}^n \max(X_i,Y_i) > 0 \\ 1 & \text{if } \sum_{i=1}^d \max(X_i,Y_i) = 0, \end{cases}$$

and the *continuous Jaccard distance* is defined as $D_c(X,Y) = 1 - J_c(X,Y)$.

In both cases the Jaccard distance is a metric. We say that the *Jaccard center* of a collection N is the set $C \subseteq U$ (resp. a non-negative real vector $C \in \mathbb{R}^n_{\geq 0}$ for the continuous case) such that $\max_{X \in N} D(X,C)$ is minimized. Throughout this paper we denote by OPT the

value of $\min_{C \subseteq U} \max_{X \in N} D(X, C)$. We always assume $\emptyset \notin N$, i.e. the empty set is not part of the input, as otherwise $\emptyset$ is a trivial optimal solution with maximum distance 1 if there exists at least one further set in N, and maximum distance 0 if $N = \{\emptyset\}$. Lastly, we will frequently use the following easily verifiable facts throughout the paper.

▶ **Fact 3.** *Let $X, Y \subseteq U$ be two item sets. Then the following statements hold:*
- $|X \cap Y| = (1 - D(X, Y)) \cdot |X \cup Y|$,
- $|X| \geq (1 - D(X, Y)) \cdot |Y|$,
- $|X \setminus Y| \leq D(X, Y) \cdot |X|$.

3 Hardness of Binary Jaccard Center

We reduce the problem of finding the optimum Jaccard center from vertex cover defined as follows.

▶ **Definition 4.** Given a graph $G(V, E)$, a vertex cover is a set $K \subset V$ such that $e \cap K \neq \emptyset$ for any $e \in E$. The minimum vertex cover is the vertex cover of smallest cardinality.

It is well known that computing the minimum vertex cover is NP-hard [17]. We will use instances with a minor constraint added for technical reasons. The minimum vertex cover will always have cardinality at most $\frac{|V|}{2} - 2$. It is easy to see that this does not affect the hardness of the vertex cover problem, for instance by adding an isolated star with one central node and $|V| + 5$ remaining nodes.

▶ **Theorem 5.** *Computing the optimum Jaccard center is NP-hard even if every $X \in N$ has cardinality at most 2.*

Proof. Let K be a minimum vertex cover of cardinality at most $\frac{|V|}{2} - 2$ in a graph $G(V, E)$ with no isolated nodes. Consider now the instance of the Jaccard center problem where the input item sets are E, the base set is V, and the center is some subset of V. We claim that a collection of vertexes C is an optimum Jaccard center if and only if C is a minimum vertex cover.

For every collection of vertices C and any edge $e \in E$, we have the following three cases:

$$D(e, C) = \begin{cases} 1 & \text{if } |C \cap e| = 0 \\ \frac{|C|}{|C|+1} & \text{if } |C \cap e| = 1 \\ \frac{|C|-2}{|C|} & \text{if } |C \cap e| = 2. \end{cases}$$

Note that the distance for some edge is 1 if and only if C is not a vertex cover. Note also that $\frac{|C|}{|C|+1} > \frac{|C|-2}{|C|}$, i.e. if $C \neq V$ then $\max_{e \in E} D(e, C) = \frac{|C|}{|C|+1}$. Now for any collection of vertices C that is a vertex cover with $|C| > |K|$, we have two cases. If $C \neq V$, then

$$\max_{e \in E} D(e, C) = \frac{|C|}{|C|+1} \geq \frac{|K|+1}{|K|+2} > \frac{|K|}{|K|+1} = \max_{e \in E} D(e, K).$$

If $C = V$, then

$$\max_{e \in E} D(e, V) = \frac{|V|-2}{|V|} = \frac{\frac{|V|}{2}-1}{\frac{|V|}{2}} \geq \frac{|K|+1}{|K|+2} > \frac{|K|}{|K|+1} = \max_{e \in E} D(e, K). \qquad \blacktriangleleft$$

▶ **Corollary 6.** *There exists no FPTAS for the binary Jaccard center problem unless P=NP.*

Proof. Two non-equal distances are at least apart by $\frac{1}{d^2}$. If an FPTAS were to exist, we could compute determine a $(1+\frac{1}{d^2})$ approximation in polynomial time. This approximation however would coincide with the optimal solution. ◀

Assuming the exponential time hypothesis (ETH), we can give stronger time bounds for PTAS and FPT. ETH, formulated by Impagliazzio, Paturi and Zane [24] assumes that there exists some positive real number s such that 3-SAT with n variables and m clauses cannot be decided in time $2^{s\cdot n}(n+m)^{O(1)}$.

▶ **Corollary 7.** *Let N be a collection of subsets over a base set U and let $C \subset U$ be the optimal Jaccard center. Assuming ETH, no FPT algorithm with parameter $k = \max_{X\in N} |C \triangle X|$, can run in time $2^{o(k)}poly(N,d)$. Further, no PTAS for the Jaccard Center problem can run in time $2^{o(\sqrt{1/\varepsilon})}poly(N,d)$.*

Proof. Under ETH, no FPT algorithm for vertex cover with parameter $|K|$, the minimal size of the vertex cover, can run in time $2^{o(|K|)}\text{poly}(N)$, see Cai and Juedes [8]. Since $k = \max_{X\in N} |C \triangle X| \in \Theta(|K|)$, the first claim follows. For the second claim, recall any PTAS approximating the Jaccard center problem beyond a factor of $(1+\frac{1}{d^2})$ recovers the optimal solution. ◀

4 Core-Covers

Our algorithms are based on the existence of a small collection M of input sets such that a high-quality center can be extracted from M. Informally, the items of an optimal center are well represented by the items of the sets contained in M. The construction is somewhat inspired by coresets for the Euclidean minimum enclosing ball problem, albeit with a weaker guarantee.

▶ **Definition 8** (Core-Covers)**.** Let N be a collection of subsets of a base set U, let OPT be the maximum distance of an optimal Jaccard center to any subset in N, and let $\alpha \geq 1$ be a parameter. A collection $M \subseteq N$ is called an *α-core-cover* if there exists an optimal center C with

$$\max_{X\in N} D\left(X, \left(\bigcup_{X\in M} X\right) \cap C\right) \leq \alpha \cdot \text{OPT}.$$

A collection $M \subseteq N$ with $A_M = \bigcap_{X\in M} X$ and $O_M = \bigcup_{X,Y\in M} X \triangle Y$ is called an *anchored α-core-cover* if there exists an optimal center C with

$$\max_{X\in N} D(X, A_M \cup (O_M \cap C)) \leq \alpha \cdot \text{OPT}.$$

We are especially interested in the size of core-covers with $\alpha = 1$ or $\alpha = 1+\varepsilon$. Core-covers are useful when the supports, i.e. the sets X are small, in which case we can find the solution by enumerating over all possible subsets of $\bigcup_{X\in M} X$. Anchored core-covers are more useful if the supports are large while the optimum value is small. For the remainder of this section, we will give (non-constructive) upper and lower bounds on the number of points required to satisfy both guarantees. Our proofs are essentially based on the following observation.

▶ **Observation 1.** *For any three sets $C, K, X \subseteq U$*

$$D(X,K) \leq D(X, K\cap C) + \frac{|K\setminus C| - 2|(X\cap K)\setminus C)|}{|X\cup K|}.$$

Proof.

$$\begin{aligned} D(X,K) &= \frac{|X \triangle K|}{|X \cup K|} = \frac{|X \triangle (K \cap C)| + |K \setminus C \setminus X| - |X \cap (K \setminus C)|}{|X \cup (K \cap C)| + |K \setminus C \setminus X|} \\ &\leq \frac{|X \triangle (K \cap C)|}{|X \cup (K \cap C)|} + \frac{|K \setminus C \setminus X| - |X \cap (K \setminus C)|}{|X \cup K|} \\ &= D(X, K \cap C) + \frac{|K \setminus C| - 2|X \cap (K \setminus C)|}{|X \cup K|} \end{aligned}$$ ◀

If X is an arbitrary input point, K is our possible solution, and C is an optimal center, this observation implies that it is sufficient to show that $D(X, K \cap C)$ is a good approximation to $D(X, C)$ and $\frac{|K \setminus C| - 2|(X \cap K) \setminus C)|}{|X \cup K|}$ is small or negative.

▶ **Lemma 9.** *For any collection of subsets N, there exists an α-core-cover M of size $\lceil 1/\varepsilon \rceil + 1$ if $\alpha = 1 + \varepsilon$ with $\varepsilon > 0$ and $\min\left\{\frac{\log(OPT \cdot |C|)}{\log(2 - OPT)} + 1, |C|\right\}$ if $\alpha = 1$.*

Proof. We show the existence of the collection M by proving that we can iteratively add a set to M such that either K is already a good approximate solution or the added set contains many elements from $C \setminus K$. Thus, finally we either have C covered by $\bigcup_{X \in M} X$ or no set violates the approximation guarantee. Let $M^{(0)} = \{X\}$ for an arbitrary $X \in N$. We denote by $K^{(i)} = C \cap \left(\bigcup_{X \in M^{(i)}} X\right)$ our solution after the i-th iteration. Note that due to Fact 3, we can assume $|C \setminus K^{(i)}| \leq \mathrm{OPT} \cdot |C|$ as $M^{(i)}$ is non-empty. In the following derivations, we assume that $\alpha \cdot \mathrm{OPT} < 1$, which is always the case for $\alpha = 1$ and always the case for $\alpha = 1 + \varepsilon$ and $\mathrm{OPT} \leq \frac{1}{1+\varepsilon}$. The latter assumption is justified by observing that otherwise any single input point already satisfies the $(1 + \varepsilon)$-core-cover guarantee.

Let $X \in N$ be a set such that $D\left(X, K^{(i)}\right) > \alpha \cdot \mathrm{OPT}$. Then

$$\begin{aligned} |X \cap (C \setminus K^{(i)})| &\overset{K^{(i)} \subseteq C}{=} |X \cap C| - |X \cap K^{(i)}| \\ &\geq (1 - \mathrm{OPT}) \cdot |X \cup C| - (1 - D(X, K^{(i)})) \cdot |X \cup K^{(i)}| \\ &> (1 - \mathrm{OPT}) \cdot |X \cup C| - \\ &\quad (1 - \alpha \cdot \mathrm{OPT}) \cdot (|X \cup C| - |C \setminus K^{(i)}| + |X \cap (C \setminus K^{(i)})|) \\ &\geq (\alpha - 1) \cdot \mathrm{OPT} \cdot |C| + \\ &\quad (1 - \alpha \cdot \mathrm{OPT}) \cdot (|C \setminus K^{(i)}| - |X \cap (C \setminus K^{(i)})|) \end{aligned}$$

For for $\alpha = 1 + \varepsilon$, we have the lower bound $|X \cap (C \setminus K^{(i)})| \geq \varepsilon \cdot \mathrm{OPT} \cdot |C|$. Since $|C \setminus K^{(0)}| \leq \mathrm{OPT} \cdot |C|$, after adding at most $s = \lceil 1/\varepsilon \rceil$ sets to $M^{(0)}$, we have $K^{(s)} = C$, or no set X with $D\left(X, K^{(s)}\right) > (1 + \varepsilon) \cdot \mathrm{OPT}$ exists.

If $\alpha = 1$, we have

$$|X \cap (C \setminus K^{(i)})| \geq \frac{1 - \mathrm{OPT}}{2 - \mathrm{OPT}} \cdot |C \setminus K^{(i)}|$$

which implies that X covers at least $\frac{1-\mathrm{OPT}}{2-\mathrm{OPT}}$ items from $C \setminus K^{(i))}$ in iteration i. Thus, $|C \setminus K^{(i)}| \leq (1 - \frac{1-\mathrm{OPT}}{2-\mathrm{OPT}})^i |C \setminus K^{(0)}| \leq (\frac{1}{2-\mathrm{OPT}})^i \cdot \mathrm{OPT} \cdot |C|$ which is smaller than 1 if $i > \frac{\log(\mathrm{OPT} \cdot |C|)}{\log(2-\mathrm{OPT})}$. Note that $|X \cap (C \setminus K^{(i)})| \geq 1$ if $D(X, K^{(i)}) > OPT$ which concludes the proof. ◀

With the space bound for core-covers, we can prove the main result of this section.

▶ **Lemma 10.** *For any collection of subsets N, there exists an anchored α-core-cover $M \subset N$ of size $O(1/\varepsilon)$ if $\alpha = 1 + \varepsilon$ with $\varepsilon > 0$ and of size $\min\{\frac{\log(OPT \cdot |C|)}{\log(2 - OPT)} + 1, |C|\} + \log \frac{OPT \cdot |C|}{1 - OPT}$ if $\alpha = 1$.*

Proof. Assume we have some optimal center C. Lemma 9 gives a set M such that $K \cap C$ is an α-approximate solution where we can represent K as $K = A_M \cup (O_M \cap C)$. Using Observation 1, the distance between K and some arbitrary set X is

$$\begin{aligned} D(X,K) &\leq D(X, K\cap C) + \frac{|K \setminus C| - 2\cdot|(X\cap K)\setminus C)|}{|X\cup K|} \\ &= D(X, K\cap C) + \frac{|A_M \setminus C| - 2\cdot|X\cap(A_M\setminus C)|}{|X\cup K|} \\ &\leq \alpha\cdot\mathrm{OPT} + \frac{|A_M \setminus C| - 2\cdot|X\cap(A_M\setminus C)|}{|X\cup K|} \end{aligned}$$

If for every $X \in N$, we have $2\cdot|X\cap(A_M\setminus C)| > |A_M\setminus C|$ then the ratio is negative and $D(X,K) \leq D(X, K\cap C) \leq \alpha\cdot\mathrm{OPT}$. Otherwise, there exists an X such that $|X\cap(A_M\setminus C)| = |(X\cap A_M)\setminus C| \leq |A_M\setminus C|/2$. We iteratively augment the collection M satisfying the space and approximation bounds of Lemma 9 with additional sets X. In each iteration, $|A_M\setminus C|$ is halved.

If $\alpha = 1$ and after adding $i > \log|A_M\setminus C|$ sets, we have $A_M\setminus C = \emptyset$. For a more precise bound on i let $Y \in M$. Then due to Fact 3,

$$|A_M\setminus C| \leq |Y\setminus C| \leq \mathrm{OPT}\cdot|Y\cup C| \leq \frac{\mathrm{OPT}\cdot|C|}{1-\mathrm{OPT}}.$$

For the case $\alpha = 1+\varepsilon$, we assume $\mathrm{OPT} < 1/(1+\varepsilon)$ as otherwise any point is a $(1+\varepsilon)$ approximation. Let $X \in N$. Again due to Fact 3 we have

$$\begin{aligned} |A_M\setminus C| &\leq \mathrm{OPT}\cdot\frac{|C|}{1-\mathrm{OPT}} \leq \mathrm{OPT}\cdot\frac{|X|}{(1-\mathrm{OPT})^2} \\ &\leq \mathrm{OPT}\cdot\frac{(1+\varepsilon)^2\cdot|X|}{\varepsilon^2} \leq \mathrm{OPT}\cdot\frac{4}{\varepsilon^2}\cdot|X|, \end{aligned}$$

where the last inequality follows for $\varepsilon \leq 1$. After adding $\log\frac{4}{\varepsilon^3}$ sets such that $|A_M\setminus C|$ is halved with each sets, we have $|A_M\setminus C|/|X\cup K| \leq \varepsilon\cdot\mathrm{OPT}\cdot|X|/|X\cup K| \leq \varepsilon\cdot\mathrm{OPT}$. Our approximation factor is therefore $\alpha\cdot\mathrm{OPT} + \varepsilon\cdot\mathrm{OPT} = (1+2\varepsilon)\cdot\mathrm{OPT}$. Rescaling ε by a factor of 2 completes the proof. ◀

We would like to remark that the bound on the number of sets required to satisfy the $(1+\varepsilon)$-core-cover guarantee is tight, and that the bound on the number of sets to satisfy the anchored $(1+\varepsilon)$-core-cover guarantee is tight up to constant multiplicative factors. Note that M is constrained to using only input sets. Better bounds are possible when we lift this restriction on M (for instance, if M consists of only an optimum center C then all guarantees are met). It is unclear whether improved guarantees not using input sets can be feasibly used in an algorithm.

▶ **Lemma 11.** *There exists a collection of subsets N such that for any $(1+\varepsilon)$-core-cover $M \subseteq N$, we have $|M| \geq 1/\varepsilon - 1$.*

Proof. For a given $\varepsilon > 0$ and assuming $1/\varepsilon$ to be an integer, we consider the following instance of vertex cover. We are given $1/\varepsilon - 1$ stars, each with at least two leaves. The optimum vertex cover and the optimum Jaccard center consists of the internal nodes, with an optimum objective value for the Jaccard center of $\frac{1/\varepsilon-1}{1/\varepsilon}$. If M does not consist of at least one edge from each star, corresponding to a set containing the element contained in the optimal Jaccard center, any center computed using only the entries of the picked edges will not intersect with at least one star, i.e. have distance 1 to the edges of the omitted star. Since $\frac{1/\varepsilon-1}{1/\varepsilon}\cdot(1+\varepsilon) = 1-\varepsilon^2 < 1$, M has to hit every star. ◀

Algorithm 1: PTAS for the Jaccard center problem

Input : Collection N of subsets, Parameter $\varepsilon > 0$
Output : $(1+\varepsilon)$-approximate Jaccard center C

1 Let $D = \{\frac{i}{j} \mid 1 \leq j \leq d \text{ and } 0 \leq i < j\}$.
2 Initialize list $C = \emptyset$.
3 **foreach** $\widehat{OPT} \in D$ **do**
4 **if** $\exists X \in N : \ \widehat{OPT} \cdot |X| < \frac{27 \ln(4n)}{\varepsilon^2}$ **then**
5 **foreach** $M \subseteq N$ with $|M| = \lceil \frac{5}{\varepsilon} + 5 \rceil$ **do**
6 Compute optimal solution $K_{\widehat{\text{OPT}}} = A_M \cup S$ with $S \subseteq O_M$ (cf. Lemma 10).
7 Add $K_{\widehat{\text{OPT}}}$ to C
8 **else**
9 Obtain non-integral solution $K'_{\widehat{\text{OPT}}}$ by solving the set of linear equations given by Equation 1
10 Obtain $K_{\widehat{\text{OPT}}}$ by rounding each entry of $K'_{\widehat{\text{OPT}}}$
11 Add $K_{\widehat{\text{OPT}}}$ to C
12 **return** $\underset{\widehat{\text{OPT}} \in D}{\text{argmin}} \{K_{\widehat{\text{OPT}}} \in C\}$

5 A PTAS for Binary Jaccard Center

This section mainly consists of the proof of the following theorem.

▶ **Theorem 12.** *Given a collection N of n subsets from a base set U of cardinality d and any $\varepsilon > 0$, there exists an algorithm computing a $(1+\varepsilon)$-approximation to the optimal Jaccard center. The algorithm runs in time $d^2 \cdot (n^{O(\varepsilon^{-6})} + LP(n,d))$, where $LP(n,d)$ is the time required to solve a linear program with n constraints and d variables.*

The algorithm (see also Algorithm 1) consists of two main steps. Let OPT be the optimal objective value. Since there are $O(d^2)$ distinct objective values for the Jaccard center problem with a base set of size d, we can try to a find solution for each value (cf. line 3 Algorithm 1). Recall that $C_i = \begin{cases} 0 & \text{if } i \notin C \\ 1 & \text{if } i \in C \end{cases}$ and that $D(X,C) \leq \text{OPT}$ holds for all $X \in N$. By multiplying both sides of the inequality with $|X \cup C|$, we obtain

$$|X \triangle C| \leq \widehat{\text{OPT}} \cdot |X \cup C|. \tag{1}$$

Observe that $|X \triangle C| = \sum_{i=1}^{d} X_i - 2X_iC_i + C_i$ and $|X \cup C| = \sum_{i=1}^{d} X_i - X_iC_i + C_i$. Hence, we obtain a set of linear inequalities which we can test for feasibility by relaxing the integrality constraints on C. Denote a feasible non-integral solution by C'. The existence of a feasible integral solution of Equation 1 implies a feasible relaxed solution C'. We interpret the C'_i as probabilities, i.e. we obtain a binary vector C by rounding each C'_i to 1 with probability C'_i. Using Chernoff bounds, this approach yields a good solution if $\text{OPT} \cdot |X| > s \cdot \log n/\varepsilon^2$ for all X and some constant s (cf. lines 4–7 of Algorithm 1).

If $\text{OPT} \cdot |Y|$ is smaller than this threshold for at least one $Y \in N$ then we could employ a naive brute force algorithm by iterating over all $\binom{d}{s \cdot \log n/\varepsilon} \in O(d^{s \cdot \log n/\varepsilon})$ subsets S and outputting the best $Y \triangle S$. To eliminate the dependency on d, we first show that a bound on $\text{OPT} \cdot |Y|$ implies that $|X_1 \triangle X_2|$ for any two sets $X_1, X_2 \in N$ is bounded. Then we compute an anchored core-cover M by enumerating all collections of $O(1/\varepsilon)$ input sets.

Having determined M, computing the optimum $A_M \cup S$ with $S \subseteq O_M$ becomes feasible (cf. lines 9–11 of Algorithm 1).

Proof of Theorem 12. In the following, we always assume that $\mathrm{OPT} < 1/(1+\varepsilon)$, as otherwise any solution is a $(1+\varepsilon)$ approximation.

To round the set of linear Equations 1, we first recall and apply the following probabilistic bounds.

▶ **Theorem 13** (Multiplicative Chernoff-Bounds [31]). *Let $B_1, \dots B_d$ be independent binary random variables with $\mu = \mathbb{E}[\sum_{i=1}^{d} B_i]$. Then for any $0 < \delta < 1$:*

$$\mathbb{P}\left[\sum_{i=1}^{d} B_i > (1+\delta)\cdot\mu\right] \leq \exp\left(-\frac{\delta^2\cdot\mu}{3}\right) \quad \textit{and} \quad \mathbb{P}\left[\sum_{i=1}^{d} B_i < (1-\delta)\cdot\mu\right] \leq \exp\left(-\frac{\delta^2\cdot\mu}{2}\right).$$

▶ **Lemma 14.** *Let S be a random binary vector obtained by rounding a fractional feasible solution of the set of Equations 1 and let $\varepsilon > 0$ be a constant. Assume that $OPT\cdot|X| \geq \frac{27\ln(4n)}{\varepsilon^2}$ for all $X \in N$. Then with probability at least $1/2$, the rounding procedure produces a binary solution S with $\max_{X\in N} D(X,S) \leq (1+\varepsilon)\cdot OPT$.*

Proof. Observe that $\mathbb{E}[|X\cup S|] \geq |X|$. We first derive concentration bounds on $|X \triangle S|$ and $|X \cup S|$. For any $X \in N$, Theorem 13 yields

$$\mathbb{P}\left[|X\cup S| < (1-\varepsilon/3)\cdot\mathbb{E}[|X\cup S|]\right] \leq \exp\left(-\frac{\varepsilon^2\cdot\mathbb{E}[|X\cup S|]}{18}\right) \leq \exp\left(-\frac{\varepsilon^2\cdot|X|}{18}\right) \leq \frac{1}{4n}$$

and

$$\begin{aligned}
& \mathbb{P}\left[|X\triangle S| > \mathbb{E}[|X\triangle S|] + \varepsilon/3\cdot\mathrm{OPT}\cdot\mathbb{E}[|X\cup S|]\right] \\
= \;& \mathbb{P}\left[|X\triangle S| > \left(1+\frac{\varepsilon\cdot\mathrm{OPT}\cdot\mathbb{E}[|X\cup S|]}{3\cdot\mathbb{E}[|X\triangle S|]}\right)\cdot\mathbb{E}[|X\triangle S|]\right] \\
\leq \;& \exp\left(-\frac{\varepsilon^2\cdot\mathrm{OPT}^2\cdot\mathbb{E}[|X\cup S|]^2}{27\cdot\mathbb{E}[|X\triangle S|]^2}\cdot\mathbb{E}[|X\triangle S|]\right) \\
\leq \;& \exp\left(-\varepsilon^2\cdot\mathrm{OPT}\cdot\mathbb{E}[|X\cup S|]/27\right) \leq \exp\left(-\varepsilon^2\cdot\mathrm{OPT}\cdot|X|/27\right) \leq \frac{1}{4n}.
\end{aligned}$$

Combining these two bounds, we have

$$\frac{|X\triangle S|}{|X\cup S|} \leq \frac{\mathbb{E}[|X\triangle S|] + \varepsilon/3\cdot\mathrm{OPT}\cdot\mathbb{E}[|X\cup S|]}{(1-\varepsilon/3)\cdot\mathbb{E}[|X\cup S|]} \leq \frac{\mathrm{OPT}+\varepsilon/3\cdot\mathrm{OPT}}{1-\varepsilon/3} \leq (1+\varepsilon)\cdot\mathrm{OPT}$$

with probability at least $1 - 1/2n$. Applying the union bound, we then obtain

$$\begin{aligned}
& \mathbb{P}\left[\max_{X\in N} D(X,S) \leq (1+\varepsilon)\cdot\mathrm{OPT}\right] \\
& = 1 - \mathbb{P}\left[\exists X\in N:\ \frac{|X\triangle S|}{|X\cup S|} > (1+\varepsilon)\cdot\mathrm{OPT}\right] \geq 1 - \frac{n}{2n} = 1/2.
\end{aligned}$$

◀

If $\mathrm{OPT}\cdot|X| > \frac{27\ln(4n)}{\varepsilon^2}$ for all $X \in N$, we can use the LP-based rounding scheme analyzed in Lemma 14 (cf. lines 4–7 of Algorithm 1). For the other cases, we will utilize Lemma 10 as follows. There exists at least one set Y with $\mathrm{OPT}\cdot|Y| \leq \frac{27\ln(4n)}{\varepsilon^2}$. With Fact 3, we have

$\mathrm{OPT} \cdot |C| \leq \mathrm{OPT} \cdot |Y|/(1-\mathrm{OPT}) \leq \frac{27\cdot(1+\varepsilon)\cdot\ln(4n)}{\varepsilon^3}$. For any two sets $X_1, X_2 \in N$, we then have

$$\begin{aligned} |X_1 \triangle X_2| &\leq 2 \cdot \mathrm{OPT} \cdot |X_1 \cup X_2| \leq 2 \cdot \mathrm{OPT} \cdot (|X_1| + |X_2|) \\ &\leq 4 \cdot \mathrm{OPT} \frac{|C|}{1-\mathrm{OPT}} \leq \frac{108 \cdot (1+\varepsilon)^2 \cdot \ln(4n)}{\varepsilon^4}. \end{aligned}$$

Let M now be a collection of sets satisfying the guarantee of Lemma 10 with $A_M = \bigcap_{X\in M} X$ and $O_M = \bigcup_{X,Y\in M} X \triangle Y$. Such a collection can be determined in time $n^{O(\varepsilon^{-1})}$ by iterating through all subsets of N of cardinality $O(\varepsilon^{-1})$. Since $|O_M| \leq \sum_{X_i\in M}\sum_{X_j\in M} |X_i \triangle X_j| \leq |M|^2 \cdot \max_{X_i,X_j\in M} |X_i \triangle X_j| \in O(\log n \cdot \varepsilon^{-6})$, we can compute an optimal solution of $\max_{X\in N} \min_{S\subseteq O_M} D(X, A_M \cup S)$ in time $2^{|O_M|} = 2^{O(\log n\cdot\varepsilon^{-6})}$.

The total running time amounts to d^2 calls to the LP given via Equations 1 or d^2 applications of Lemma 10 with a running time of $2^{O(\log n\cdot\varepsilon^{-6})} = n^{O(\varepsilon^{-6})}$. ◀

6 An FPT Algorithm for Binary Jaccard Center

Our second application of core-covers is an FPT algorithm in the parameter $k = \max_{X\in N} |X \triangle C|$ where C is an arbitrary optimal solution. The main technical difficulty is to efficiently construct a core-cover without enumerating all possible core-covers. We first bound the size of an anchored 1-core-cover given by Lemma 10 in terms of k.

▶ **Lemma 15.** *For any collection N of subsets and an optimal center C with cost $OPT < 1$, let $k = \max_{X\in N} |X \triangle C|$. Then*

$$\min\left\{\frac{\log(OPT\cdot|C|)}{\log(2-OPT)} + 1, |C|\right\} \leq 2k \text{ and } \log\frac{OPT\cdot|C|}{1-OPT} \leq 3\log k.$$

Proof. There exists an $X \in N$ such that $k \geq |X \triangle C| = \mathrm{OPT} \cdot |X \cup C| \geq \mathrm{OPT} \cdot |C|$. We first note that both terms are increasing with OPT, hence we assume $\mathrm{OPT} > 1/2$. Then $|X \triangle C|/|X \cup C| = \mathrm{OPT}$ for some $X \in N$ implying

$$(1-\mathrm{OPT}) = \mathrm{OPT} \cdot |X \cap C|/|X \triangle C| \geq \frac{1}{2|X \triangle C|} \geq \frac{1}{2k}.$$

Therefore, we have $1/(1-\mathrm{OPT})) \leq 2k$,

$$\log(2-\mathrm{OPT}) = \log(1+1-\mathrm{OPT}) = \frac{\ln(1+1-\mathrm{OPT})}{\ln 2} \geq \frac{1-\mathrm{OPT}}{2\ln 2} \geq \frac{1}{4k\ln 2},$$

and

$$\min\left\{\frac{\log(\mathrm{OPT}\cdot|C|)}{\log(2-\mathrm{OPT})} + 1, |C|\right\} \leq |C| \leq 2k \text{ and } \log\frac{\mathrm{OPT}\cdot|C|}{1-\mathrm{OPT}} \leq 1 + 2\log k.$$ ◀

For a given estimate of OPT, the algorithm initially chooses two arbitrary sets to be included in the anchored core-cover M. If the optimal solution $A_M \cup S$ with $S \subseteq O_M$ satisfies $\max_{X\in N} D(X, A_M \cup S) < \mathrm{OPT}$ then we can reduce our estimate of OPT. Otherwise, we add any set X at distance greater than OPT to M. The set X improves the core-cover, either by increasing $|C \cap (A_M \cup O_M)|$ or by decreasing $|A_M \setminus C|$ for some optimal center C. Lemma 15 allows us to bound the number of times this happens before M satisfies the anchored core-cover guarantee, upon which we can recover the optimum solution.

Algorithm 2: FPT-algorithm for the Jaccard center problem

Input : Collection N of subsets, Parameter $k = \max_{X \in N} |X \triangle C|$
Output : Optimal Jaccard center C

1 Let $D = \{\frac{i}{j} \mid 1 \leq j \leq d \text{ and } 0 \leq i < j\}$.
2 Initialize list $C = \emptyset$.
3 **foreach** $\widehat{OPT} \in D$ **do**
4 Initialize $M = \{X, Y\}$ with arbitrary $X, Y \in N$ and $X \neq Y$.
5 **for** $i = 1$ **to** $5k$ **do**
6 Compute optimal solution $K_{\widehat{\text{OPT}}} = A_M \cup S$ with $S \subseteq O_M$ (cf. Lemma 10).
7 **if** $\exists X \in N : D(X, K_{\widehat{OPT}}) > \widehat{OPT}$ **then**
8 $M = M \cup \{X\}$
9 **else**
10 Add $K_{\widehat{\text{OPT}}}$ to C
11 **break**
12 **return** $\operatorname{argmin}_{\widehat{\text{OPT}} \in D} \{K_{\widehat{\text{OPT}}} \in C\}$

▶ **Theorem 16.** *Algorithm 2 computes an optimal Jaccard center C satisfying $\max_{X \in N} |X \triangle C| = k$ in time $2^{O(k^3)} \cdot n \cdot d^3$.*

Proof. Let $\widehat{\text{OPT}} \in D$ be a guess for our optimal value OPT. If $\widehat{\text{OPT}} < \text{OPT}$ then the loop terminates without finding a center. Let $\widehat{\text{OPT}} \geq \text{OPT}$. Using Observation 1, we know that

$$D(X, K_{\widehat{\text{OPT}}}) \leq D(X, K_{\widehat{\text{OPT}}} \cap C) + \frac{|K_{\widehat{\text{OPT}}} \setminus C| - 2 \cdot |(X \cap K_{\widehat{\text{OPT}}}) \setminus C)|}{|X \cup K_{\widehat{\text{OPT}}}|}.$$

If $D(X, K_{\widehat{\text{OPT}}}) > \widehat{\text{OPT}}$ then we distinguish between two cases:

Case $|K_{\widehat{\text{OPT}}} \setminus C| - 2 \cdot |(X \cap K_{\widehat{\text{OPT}}}) \setminus C)| \leq 0$:
Then $D(X, K_{\widehat{\text{OPT}}} \cap C) > \widehat{\text{OPT}}$ and we can apply the analysis of Lemma 9. Thus, we add at most $2k$ sets to M until this case can no longer occur (Lemma 15).

Case $|K_{\widehat{\text{OPT}}} \setminus C| - 2 \cdot |(X \cap K_{\widehat{\text{OPT}}}) \setminus C)| > 0$:
Then $|(X \cap K_{\widehat{\text{OPT}}}) \setminus C)| \leq |K_{\widehat{\text{OPT}}} \setminus C|/2$ and we can apply the analysis of Lemma 10. Thus, we add at most $3 \log k$ points to M until this case can no longer occur (Lemma 15).

The computation of $K_{\widehat{\text{OPT}}}$ can be done in time $2^{O(k^3)}$ by an exhaustive search over all possible subsets of O_M since

$$|O_M| \leq |M^2| \cdot \max_{X,Y \in N} |X \triangle Y| \leq O(k^2) \cdot \max_{X,Y \in N} (|X \triangle C| + |Y \triangle C|) = O(k^3).$$

We perform the exhaustive search $O(k)$ times and for each solution we evaluate the objective value for each set. Since $|D| = O(d^2)$ and we examine every set in line 7 of Algorithm 2, the algorithm terminates in time $2^{O(k^3)} \cdot n \cdot d^3$. ◀

7 A Note on Continuous Jaccard Center

We conclude by briefly describing how to find the continuous Jaccard center. We will formulate the decision problem of finding a center with distance at most *dist* as an LP. The

optimum center can thereafter be determined in polynomial time using binary search over the possible values of $dist$. In the following let $X^j \in N$ be the jth point of N w.r.t. some arbitrary ordering. We use the variable $c_i \geq 0$ to denote the ith entry of the Jaccard center. We further use the variables $a_{i,j}$ and $b_{i,j}$ for all $i \in \{1, \ldots, d\}$ and $j \in \{1, \ldots n\}$ to denote the maximum and minimum of X_i^j and c_i. We then use the constraints

$$\sum_{i=1}^{d} b_{i,j} \geq (1 - dist) \cdot \sum_{i=1}^{d} a_{i,j} \qquad \text{for all } j \in \{1, \ldots n\}$$

$$b_{i,j} \leq c_i, X_i^j \leq a_{i,j} \qquad \text{for all } j \in \{1, \ldots n\}, i \in \{1, \ldots d\}$$

$$a_{i,j}, b_{i,j}, c_i \geq 0 \qquad \text{for all } j \in \{1, \ldots n\}, i \in \{1, \ldots d\}.$$

Note that the top most equation $\sum_{i=1}^{d} \min(c_i, X_i^j) \geq (1 - dist) \cdot \sum_{i=1}^{d} \max(c_i, X_i^j)$ is equal to $1 - \frac{\sum_{i=1}^{d} \min(X_i, Y_i)}{\sum_{i=1}^{d} \max(X_i, Y_i)} \leq dist$.

References

1. P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.
2. C. Bachmaier, F. J. Brandenburg, A. Gleißner, and A. Hofmeier. On the hardness of maximum rank aggregation problems. *J. Discrete Algorithms*, 31:2–13, 2015.
3. M. Badoiu and K. L. Clarkson. Optimal core-sets for balls. *Comput. Geom.*, 40(1):14–22, 2008.
4. M. Badoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 250–257, 2002.
5. T. C. Biedl, F.-J. Brandenburg, and X. Deng. On the complexity of crossings in permutations. *Discrete Mathematics*, 309(7):1813–1823, 2009.
6. A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES'97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society.
7. A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.
8. L. Cai and D. W. Juedes. On the existence of subexponential parameterized algorithms. *J. Comput. Syst. Sci.*, 67(4):789–807, 2003.
9. F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 – July 1, 2009*, pages 219–228, 2009.
10. F. Chierichetti, R. Kumar, S. Pandey, and S. Vassilvitskii. Finding the Jaccard median. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 293–311, 2010.
11. K. L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms*, 6(4), 2010.
12. E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Trans. Knowl. Data Eng.*, 13(1):64–78, 2001.
13. A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 271–280, 2007.

14 C. de la Higuera and F. Casacuberta. Topology of strings: Median string is NP-complete. *Theor. Comput. Sci.*, 230(1-2):39–48, 2000.

15 M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of CLOSEST substringsize and related problems. In *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes – Juan les Pins, France, March 14-16, 2002, Proceedings*, pages 262–273, 2002.

16 M. Frances and A. Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997.

17 M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

18 L. Gasieniec, J. Jansson, and A. Lingas. Efficient approximation algorithms for the Hamming center problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland.*, pages 905–906, 1999.

19 T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

20 J. C. Gower and P. Legendre. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48, 1986.

21 J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica*, 37(1):25–42, 2003.

22 S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Inf. Syst.*, 25(5):345–366, 2000.

23 D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33(3):533–550, 1986.

24 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

25 P. Jaccard. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

26 P. Kumar, J. S. B. Mitchell, and E. A. Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM Journal of Experimental Algorithmics*, 8, 2003.

27 J. K. Lanctôt, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003.

28 M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002.

29 D. Marx. Closest substring problems with small distances. *SIAM J. Comput.*, 38(4):1382–1410, 2008.

30 N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984.

31 M. Mitzenmacher and E. Upfal. *Probability and computing – randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

32 F. Nicolas and E. Rivals. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *J. Discrete Algorithms*, 3(2-4):390–415, 2005.

33 V. Y. Popov. Multiple genome rearrangement by swaps and by element duplications. *Theor. Comput. Sci.*, 385(1-3):115–126, 2007.

34 R. Real and J. M. Vargas. The probabilistic basis of jaccard's index of similarity. *Systematic biology*, 45(3):380–385, 1996.

35 H. Späth. The minisum location problem for the Jaccard metric. *Operations-Research-Spektrum*, 3(2):91–94, 1981.

36 J. J. Sylvester. A Question in the Geometry of Situation. *Quarterly Journal of Pure and Applied Mathematics*, 1, 1857.

37 G. A. Watson. An algorithm for the single facility location problem using the Jaccard metric. *SIAM Journal on Scientific and Statistical Computing*, 4(4):748–756, 1983.

38 E. Welzl. *Smallest enclosing disks (balls and ellipsoids)*, pages 359–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

39 P. Willett, J. M. Barnard, and G. M. Downs. Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, 38(6):983–996, 1998.

40 E. A. Yildirim. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization*, 19(3):1368–1391, 2008.

The Polytope-Collision Problem*

Shaull Almagor[†1], Joël Ouaknine[‡2], and James Worrell[§3]

1 **Department of Computer Science, Oxford University, Oxford, UK**
shaull.almagor@cs.ox.ac.uk
2 **Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany**
joel@mpi-sws.org
3 **Department of Computer Science, Oxford University, Oxford, UK**
jbw@cs.ox.ac.uk

Abstract

The *Orbit Problem* consists of determining, given a matrix $\mathcal{A} \in \mathbb{R}^{d\times d}$ and vectors $x, y \in \mathbb{R}^d$, whether there exists $n \in \mathbb{N}$ such that $\mathcal{A}^n = y$. This problem was shown to be decidable in a seminal work of Kannan and Lipton in the 1980s. Subsequently, Kannan and Lipton noted that the Orbit Problem becomes considerably harder when the *target* y is replaced with a subspace of $\mathbb{R}^d$. Recently, it was shown that the problem is decidable for vector-space targets of dimension at most three, followed by another development showing that the problem is in **PSPACE** for polytope targets of dimension at most three.

In this work, we take a dual look at the problem, and consider the case where the *initial* vector x is replaced with a polytope P_1, and the target is a polytope P_2. Then, the question is whether there exists $n \in \mathbb{N}$ such that $\mathcal{A}^n P_1 \cap P_2 \neq \emptyset$. We show that the problem can be decided in **PSPACE** for dimension at most three. As in previous works, decidability in the case of higher dimensions is left open, as the problem is known to be hard for long-standing number-theoretic open problems.

Our proof begins by formulating the problem as the satisfiability of a parametrized family of sentences in the existential first-order theory of real-closed fields. Then, after removing quantifiers, we are left with instances of simultaneous positivity of sums of exponentials. Using techniques from transcendental number theory, and separation bounds on algebraic numbers, we are able to solve such instances in **PSPACE**.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, I.1.2 Algorithms

Keywords and phrases linear dynamical systems, orbit problem, algebraic algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.24

1 Introduction

Given a linear transformation $\mathcal{A}$ over the vector space $\mathbb{R}^d$, together with a starting point x, the *orbit* of x under $\mathcal{A}$ is the infinite sequence $x, \mathcal{A}x, \mathcal{A}^2x, \ldots$. A natural decision problem in discrete linear dynamical systems is whether the orbit of x ever hits a particular target set V (assuming suitable, effective representations of $\mathcal{A}$, x, and V). An early instance of

* The full version of the paper can be found at https://arxiv.org/abs/1611.01344.
† Shaull Almagor is supported by ERC grant AVS-ISS (648701).
‡ Joël Ouaknine is also affiliated with the Department of Computer Science, Oxford University, UK, and is supported by ERC grant AVS-ISS (648701).
§ James Worrell is supported by EPSRC grant EP/N008197/1.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 24; pp. 24:1–24:14


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

this problem was raised by Harrison in 1969 [12] for the special case in which V is simply a point in $\mathbb{R}^d$. Decidability remained open for over ten years, and was finally settled in a seminal paper of Kannan and Lipton, who moreover gave a polynomial-time decision procedure [13]. In subsequent work [14], Kannan and Lipton noted that the Orbit Problem becomes considerably harder when the target V is replaced by a subspace of $\mathbb{R}^d$: indeed, if V has dimension $d-1$, the problem is equivalent to the *Skolem Problem*, known to be **NP-Hard** but whose decidability has remained open for over 80 years [21]. However, for low-dimensional target spaces, the Orbit Problem becomes more tractable. Indeed, it was recently shown in [7] that the problem is decidable for vector-space targets of dimension at most three, with polynomial-time complexity for one-dimensional targets, and complexity in $\mathbf{NP}^{\mathbf{RP}}$ for two- and three-dimensional targets. Another development followed in [8], where the authors consider more intricate target sets, namely polytopes. It is shown in [8] that up to dimension three, the problem can be solved in **PSPACE**. In addition, it is shown that for higher dimensions, the problem becomes hard with respect to long-standing number-theoretic open problems.

A key motivation for studying the Orbit Problem comes from program verification, particularly the problem of determining whether a simple while loop with affine assignments and guards will terminate or not. Similar reachability questions were considered and left open by Lee and Yannakakis in [15] for what they termed "real affine transition systems". Similarly, decidability for the case of a single-halfspace target was mentioned as an open problem by Braverman in [5].

An important aspect of termination problems for linear loops is the quantification of the initial point. Traditionally, the 'Termination problem' in the program-verification literature (see, e.g. [4]) refers to termination of while loops for *all* possible initial starting points. In [17] the traditional Termination Problem is solved over the integers for while loops, assuming diagonalisability of the associated linear transformation. To our knowledge, very little else is known on the general problem of universally quantified inputs. In contrast, the works in [7, 8] study the termination problem where the input is fixed (but the target space is complicated). This corresponds to verifying the termination of a concrete run of a linear loop. It should be noted that the techniques used for analyzing the latter differ significantly from the former.

In this work, we take a dual look at the problem, and study the case where the input is existentially quantified. Thus, we are given a set $P_1 \subseteq \mathbb{R}^d$, and a target set P_2, and the problem is to decide whether there exists $x \in P_1$ and $n \in \mathbb{N}$ such that $\mathcal{A}^n x \in P_2$. In practice, this corresponds to deciding *safety* properties of linear loops: we think of P_2 as some error set, and the problem is to decide whether there exists an input that would cause the program to reach the error set.

Specifically, the focus of this paper is the **3D Polytope-Collision Problem** (**3DPCP**, for short): Given two polytopes P_1 and P_2 in $\mathbb{R}^3$ (represented as an intersection of halfspaces) and a matrix with real-algebraic entries[1] $\mathcal{A} \in (\mathbb{A} \cap \mathbb{R})^{3\times 3}$, determine whether there exists a point $x \in P_1$ and a natural number n such that $\mathcal{A}^n x \in P_2$.

We present the following effectiveness result on the 3D Polytope-Collision Problem.

▶ **Theorem 1.** *3DPCP is decidable in* ***PSPACE****.*

Note that as proved in [8], when the dimension is at least four, the polytope-collision problem becomes hard with respect to number-theoretic open problems.

[1] We denote by $\mathbb{A}$ the set of algebraic numbers.

Before describing our approach, we explain why this result is somewhat surprising. Consider a simplification of 3DPCP, where the initial polytope P is a segment between points x and y, and we wish to decide whether the orbit of P under the matrix $\mathcal{A}$ collides with another polytope R. We can represent P as the single point (x, y) in $\mathbb{R}^6$, and extend $\mathcal{A}$ to a matrix $\mathcal{B} \in \mathbb{R}^{6\times 6}$ that has two copies of $\mathcal{A}$ on its diagonal. Then, the orbit of P under $\mathcal{A}$ corresponds to the orbit of (x, y) under $\mathcal{B}$. However, the respective target space in $\mathbb{R}^6$ becomes the set of all points (u, v) such that the line between u and v in $\mathbb{R}^3$ intersects R. While this is a semi-algebraic set, it is quite complicated, and recall that the polytope hitting problem is already hard in dimension four. Thus, this approach suggests that the problem may be as hard as the hitting problem in $\mathbb{R}^6$.

Technically, the above intricacy prevents us from using the techniques previously employed on fixed-input orbit problems, e.g. [8]. There, describing the dominant behavior of the orbit is relatively straightforward, and the difficulty is reasoning about hitting the target. In our setting, merely describing the orbit involves symbolic quantifier elimination, as described next, and reasoning about hitting the target therefore involves symbolic analysis.

Our approach to proving Theorem 1 is as follows. Observe that 3DPCP can be formulated as the problem of deciding whether there exists $n \in \mathbb{N}$ such that $\mathcal{A}^n P_1$ intersects P_2 (where $\mathcal{A}^n P_1 = \{\mathcal{A}^n x : x \in P_1\}$). In Section 3 we reduce this formulation of 3DPCP to the problem of solving a system of inequalities, as we now describe.

In Section 3.1 we identify two types of intersection of 3D polytopes, namely (1) where a vertex of one polytope lies in the other polytope, and (2) where an edge of one polytope intersects a face of the other polytope. We show that under a certain representation, an intersection of polytopes is always of one of these types. Note that while each of these types seems symmetric with respect to the two polytopes, in our setting the polytopes have an inherent asymmetry, as $\mathcal{A}^n P_1$ is dependent on n whereas P_2 is not.

In order to overcome this asymmetry, in Section 3.2 we reduce 3DPCP to the case where the matrix $\mathcal{A}$ is invertible. Then, considering $\mathcal{A}^n P_1$ and P_2 is symmetric to considering P_1 and $(\mathcal{A}^{-1})^n P_2$.

Next, in Section 3.3 we observe that intersections of Type (1) can be decided using the work in [8], and we are left to address intersections of Type (2). We formulate this type of intersection as $\exists n \in \mathbb{N}\ \Phi(\alpha^n, \overline{\alpha}^n, \rho^n)$, where Φ is a sentence in the existential first-order theory of real-closed fields, and $\alpha, \overline{\alpha}$, and ρ are the eigenvalues of the matrix $\mathcal{A}$, with $\alpha \in \mathbb{A}\setminus\mathbb{R}$ and $\rho \in \mathbb{A} \cap \mathbb{R}$ (the case where $\mathcal{A}$ has only real eigenvalues is simpler, and we handle it in the full version). Moreover, Φ contains only linear expressions (with respect to its variables, where n is treated as a constant), and at most three real variables. We proceed by eliminating the quantifiers from Φ. We use the fact that the expressions in $\Phi(n)$ are linear to apply the simple Fourier-Motzkin quantifier-elimination algorithm [11]. We note that while other quantifier-elimination algorithms (e.g., [20]) offer better asymptotic complexity, since the number of variables in Φ is constant, Fourier-Motzkin elimination takes polynomial time. Moreover, its simplicity allows us to keep track of the expressions in the quantifier free equivalent of $\Phi(n)$. Specifically, we show that this output consists of a disjunction of systems, where each system is a conjunction of expressions of the form

$$A\alpha^{2n} + \overline{A}\overline{\alpha}^{2n} + B\alpha^n\rho^n + \overline{B}\overline{\alpha}^n\rho^n + C\rho^{2n} + D|\alpha|^{2n} + E\alpha^n + \overline{E}\overline{\alpha}^n + F\rho^n + G \bowtie 0 \qquad (1)$$

where $\bowtie \in \{>, =\}$.

Finally, Section 4 is the heart of our technical contribution, in which we show how to solve such systems. Intuitively, we normalize Expression (1) such that the maximal modulus of its terms is 1, thus obtaining an expression of the form $A\gamma^{2n} + \overline{A}\overline{\gamma}^{2n} + B\gamma^n + \overline{B}\overline{\gamma}^n + C + r(n) \bowtie 0$

with $|\gamma| = 1$ and $r(n)$ tending exponentially fast to 0. We then consider two cases, depending on whether γ is a root of unity or not. If γ is a root of unity, we show that it is enough to consider polynomially many expressions with only real elements, which can be handled using relatively standard techniques. If γ is not a root of unity, things are more involved. Then, by utilizing consequences of the Baker-Wüstholz theorem [2], we are able to show that the expression $|A\gamma^{2n} + \overline{A}\overline{\gamma}^{2n} + B\gamma^n + \overline{B}\overline{\gamma}^n + C|$ is bounded away from 0 by an inverse polynomial in n. Then, using a separation bound due to Mignotte [16], we show that $r(n)$ decays fast enough to obtain a bound $N \in \mathbb{N}$ such that $r(n)$ does not affect the sign of $A\gamma^{2n} + \overline{A}\overline{\gamma}^{2n} + B\gamma^n + \overline{B}\overline{\gamma}^n + C$ for all $n > \mathbb{N}$. Finally, since γ is not a root of unity, it is dense in the unit circle, and we can replace the analysis of the former expression by analysis of the simpler function $f(z) = Az^2 + \overline{A}\overline{z}^2 + Bz + \overline{B}\overline{z} + C$ on the unity circle, from which we obtain our main result.

2 Mathematical Tools

In this section we introduce the key technical tools used in this paper.

2.1 Algebraic numbers

For $p \in \mathbb{Z}[x]$ a polynomial with integer coefficients we denote by $\|p\|$ the bit length of its representation as a list of coefficients encoded in binary. Note that the *degree* of p, denoted $\deg(p)$ is at most $\|p\|$, and the *height* of p – i.e., the maximum of the absolute values of its coefficients, denoted $H(p)$ – is at most $2^{\|p\|}$.

We begin by summarising some basic facts about algebraic numbers (denoted $\mathbb{A}$) and their (efficient) manipulation. The main references include [3, 9, 20]. A complex number α is *algebraic* if it is a root of a single-variable polynomial with integer coefficients. The *defining polynomial* of α, denoted p_α, is the unique polynomial of least degree, and whose coefficients do not have common factors, which vanishes at α. The *degree* and *height* of α are respectively those of p, and are denoted $\deg(\alpha)$ and $H(\alpha)$. A standard representation[2] for algebraic numbers is to encode α as a tuple comprising its defining polynomial together with rational approximations of its real and imaginary parts of sufficient precision to distinguish α from the other roots of p_α. More precisely, α can be represented by $(p_\alpha, a, b, r) \in \mathbb{Z}[x] \times \mathbb{Q}^3$ provided that α is the unique root of p_α inside the circle in $\mathbb{C}$ of radius r centred at $a + bi$. A separation bound due to Mignotte [16] asserts that for roots $\alpha \neq \beta$ of a polynomial $p \in \mathbb{Z}[x]$, we have

$$|\alpha - \beta| > \frac{\sqrt{6}}{d^{(d+1)/2} H^{d-1}} \tag{2}$$

where $d = \deg(p)$ and $H = H(p)$. Thus if r is required to be less than a quarter of the root-separation bound, the representation is well-defined and allows for equality checking. Given a polynomial $p \in \mathbb{Z}[x]$, it is well-known how to compute standard representations of each of its roots in time polynomial in $\|p\|$ [3, 9, 19]. Thus given an algebraic number α for which we have (or wish to compute) a standard representation, we write $\|\alpha\|$ to denote the bit length of this representation. From now on, when referring to computations on algebraic numbers, we always implicitly refer to their standard representations.

Note that Equation 2 can be used more generally to separate arbitrary algebraic numbers: indeed, two algebraic numbers α and β are always roots of the polynomial $p_\alpha p_\beta$ of degree

[2] Note that this representation is not unique.

at most $\deg(\alpha) + \deg(\beta)$, and of height at most $H(\alpha)H(\beta)$. Given algebraic numbers α and β, one can compute $\alpha + \beta$, $\alpha\beta$, $1/\alpha$ (for $\alpha \neq 0$), $\overline{\alpha}$, and $|\alpha|$, all of which are algebraic, in time polynomial in $\|\alpha\| + \|\beta\|$. Likewise, it is straightforward to check whether $\alpha = \beta$. Moreover, if $\alpha \in \mathbb{R}$, deciding whether $\alpha > 0$ can be done in time polynomial in $\|\alpha\|$. Efficient algorithms for all these tasks can be found in [3, 9].

2.2 First-order theory of the reals

Let $\overrightarrow{x} = x_1, \ldots, x_m$ be a list of m real-valued variables, and let $\sigma(\overrightarrow{x})$ be a Boolean combination of atomic predicates of the form $g(\overrightarrow{x}) \bowtie 0$, where each $g(\overrightarrow{x}) \in \mathbb{Z}[x]$ is a polynomial with integer coefficients over these variables, and $\bowtie \in \{>, =\}$. A *sentence of the first-order theory of the reals* is of the form $Q_1 x_1 Q_2 x_2 \cdots Q_m x_m \sigma(\overrightarrow{x})$, where each Q_i is one of the quantifiers $\exists$ or $\forall$. Let us denote the above formula by τ , and write $\|\tau\|$ to denote the bit length of its syntactic representation. Tarski famously showed that the first-order theory of the reals is decidable [22]. His procedure, however, has non-elementary complexity. Many substantial improvements followed over the years, starting with Collins' technique of cylindrical algebraic decomposition [10], and culminating with the fine-grained analysis of Renegar [20]. In this paper, we focus exclusively on the situation in which the number of variables is uniformly bounded.

▶ **Theorem 2** (Renegar). *Let $M \in \mathbb{N}$ be fixed, let τ be of the form $Q_1 x_1 Q_2 x_2 \cdots Q_m x_m \sigma(\overrightarrow{x})$. Assume that the number of variables in τ is bounded by M (i.e., $m \leq M$). Then the truth value of τ can be determined in time polynomial in $\|\tau\|$.*

An important property of the first-order theory of the reals is that it admits *quantifier elimination*. That is, consider two lists of variables $\overrightarrow{x}$, $\overrightarrow{y}$ and a sentence $Q_1 x_1 \cdots Q_m x_m \sigma(\overrightarrow{x}, \overrightarrow{y})$ with the variables of $\overrightarrow{y}$ being free, then there exists an (unquantified) sentence $\sigma'(\overrightarrow{y})$ such that for every assignment π to the variables in $\overrightarrow{y}$ it holds that $\sigma'(\pi)$ is true iff $Q_1 x_1 \cdots Q_m x_m \sigma(\overrightarrow{x}, \pi)$ is true.

When the polynomials in σ are all linear and the quantifiers are all existential, then quantifier elimination can be performed using the Fourier-Motzkin quantifier-elimination algorithm [11] (see the full version for details). The benefit of this algorithm is its simplicity, which allows us to remove quantifiers symbolically.

We remark that algebraic constants can also be incorporated as coefficients in the first-order theory of the reals, as follows. Consider a polynomial $g(x_1, \ldots, x_m)$ with algebraic coefficients $c_1, \ldots, c_k$. We replace every c_i with a new, existentially-quantified variable y_i, and add to the sentence the predicates $p_{c_i}(y_i) = 0$ and $(y_i - (a + bi))^2 < r^2$, where (p_{c_i}, a, b, r) is the representation of c_i. Then, in any evaluation of this formula to True, it must hold that y_i is assigned value c_i.

2.3 Polytopes and their representation

A *polytope* P in $\mathbb{R}^3$ is an intersection of finitely many halfspaces in $\mathbb{R}^3$: $P = \{x \in \mathbb{R}^3 : v_1^T x \geq c_1 \wedge \ldots \wedge v_k^T x \geq c_k\}$ for vectors $v_1, \ldots, v_k \in \mathbb{R}^3$ and numbers $c_1, \ldots, c_k \in \mathbb{R}$. The *halfspace description* of P is then $(v_1, c_1), \ldots, (v_k, c_k)$. When all entries are algebraic, we denote by $\|P\|$ the description length.

The *dimension* of a polytope P, denoted $\dim(P)$, is the dimension of the subspace of $\mathbb{R}^3$ spanned by P. The dimension of P can be computed in time polynomial in $\|P\|$ by solving polynomially many linear programs. In $\mathbb{R}^3$, the dimension of a polytope is in $\{0, \ldots, 3\}$. A 2D boundary of a 3D polytope is a 2D polytope called a *face*. Similarly, the boundries of 2D

polytopes (and in particular of faces) are called *edges*, and the boundries of edges are *vertices*. Every 3D polytope, except the trivial $\mathbb{R}^3$ and $\emptyset$, has at least one face (but not necessarily edges or vertices). Since vertices and edges are crucial for our algorithms, we present the following lemma from [8].

▶ **Lemma 3** ([8] Lemma A.1). *Suppose $P \subseteq \mathbb{R}^3$ is a 2D polytope. Then $P = \bigcup_{i=1}^{m} A_i$, where m is finite and each A_i is of the form $A_i = \{u_i + \alpha v_i + \beta w_i : T_i(\alpha, \beta)\}$ where $u_i, v_i, w_i \in \mathbb{R}^3$ and the predicates $T_i(\alpha, \beta)$ are from the following:*

- $T_i(\alpha, \beta) \equiv \alpha \geq 0 \wedge \beta \geq 0$ *(A_i is an infinite cone)*
- $T_i(\alpha, \beta) \equiv \alpha \geq 0 \wedge \beta \geq 0 \wedge \alpha + \beta \leq 1$ *(A_i is a triangle)*
- $T_i(\alpha, \beta) \equiv \alpha \geq 0 \wedge \beta \geq 0 \wedge \beta \leq 1$ *(A_i is an infinite strip)*

Furthermore, if we are given a halfspace description of P with length $\|P\|$, the size of the representation of each vector u_i, v_i, w_i is at most $\|P\|^{O(1)}$.

Note that since the representation of u_i, v_i, and w_i is polynomial, it follows that m is at most exponential in $\|P\|$, and moreover, that iterating over the sets A_i can be done in **PSPACE**.

3 From 3DPCP to a System of Inequalities

In this section we reduce 3DPCP to the problem of solving a system of inequalities. More precisely, we show how to solve 3DPCP by solving an exponential number of systems of equalities and inequalities, and that iterating over these systems can be done in **PSPACE**. In Section 4 we tackle the main technical challenge of solving each such system in **PSPACE**, thus concluding the proof of Theorem 1.

As mentioned in Section 1, we start by studying the intersection of polytopes.

3.1 Intersection of polytopes

Consider two intersecting polytopes Q_1 and Q_2 in $\mathbb{R}^3$. In this section, we characterize the intersection of Q_1 and Q_2, which would later simplify the solution of 3DPCP. To illustrate the idea, assume that both Q_1 and Q_2 are bounded 3D polytopes. In this case, we can assume w.l.o.g. that Q_1 and Q_2 are both tetrahedra. Indeed, every bounded 3D polytope with d vertices can be decomposed into a union of at most $\binom{d}{4}$ tetrahedra, and two such decompositions intersect iff two of the tetrahedra in the respective decompositions intersect. Under this assumption, there are two possible "types" of intersections: either Q_1 is contained in Q_2 (or vice-versa), or an edge of Q_1 intersects a face of Q_2 (or vice-versa). When the polytopes are bounded, we can relax the first requirement, and require instead that a vertex of Q_1 lies in Q_2 (or vice-versa).

In general, however, Q_1 or Q_2 may be unbounded. In this case we need to be slightly more careful. Indeed, as stated in Section 2.3, unbounded polytopes might have no vertices or edges, but only faces (unless the polytope is $\mathbb{R}^3$ or $\emptyset$, in which case the problem is trivial). For example, consider the case where Q_1 and Q_2 are infinite prisms. Then, it is possible that $Q_1 \cap Q_2 \neq \emptyset$ and neither are contained in each other, but no edge of Q_1 intersects a face of Q_2 (and vice-versa).

Therefore, to get the above characterization for unbounded polytopes, we need to add "fictive" edges. Since we assume the input polytopes are non trivial, then each of them has at least one face, and recall that the faces of a 3D polytope are 2D polytopes. By employing Lemma 3 on the faces of the polytopes, we get that each face of Q_1 and of Q_2 can be written as $\bigcup_{i=1}^{m} A_i$ as per Lemma 3. Observe that every set A_i in the decomposition of Lemma 3

has at least two edges and one vertex, and that a non-empty intersection $A_i \cap A'_j$ in such decompositions also intersects an edge of at least one of the two sets (the only involved case is the intersection of two infinite strips, where one should notice that the strips are only infinite to one side).

We conclude that the above characterization of the intersection of polytopes is correct also for unbounded ones. In the following, when we refer to a vertex/edge of an unbounded polytope, we mean the vertices and edges of the sets in the decomposition of Lemma 3.

Thus, we have that Q_1 intersects Q_2 if at least one of the following holds:

1. There exists a vertex of Q_1 that is in Q_2.
2. There exists a vertex of Q_2 that is in Q_1.
3. An edge of Q_1 intersects a face of Q_2.
4. An edge of Q_2 intersects a face of Q_1.

3.2 Reduction to the invertible case

In the notations of Section 3.1, we wish to check the intersection of $Q_1 = \mathcal{A}^n P_1$ and $Q_2 = P_2$ for an existentially quantified $n \in \mathbb{N}$. As mentioned in Section 1, if $\mathcal{A}$ is invertible, then the problem is symmetric with respect to Q_1 and Q_2. Indeed, $\mathcal{A}^n P_1$ intersects P_2 iff P_1 intersects $(\mathcal{A}^{-1})^n P_2$. However, if $\mathcal{A}$ is not invertible, the problem is not clearly symmetric. In this section, we reduce 3DPCP to the case where $\mathcal{A}$ is an invertible matrix.

Consider polytopes $P, R \subseteq \mathbb{R}^3$, and let $\mathcal{A} \in (\mathbb{A} \cap \mathbb{R})^{3\times 3}$ be a singular matrix, so 0 is an eigenvalue of $\mathcal{A}$. Consider first the case where the multiplicity of 0 is 1. Thus, we can write $\mathcal{A} = D^{-1} \begin{pmatrix} 0 & 0 \\ 0 & B \end{pmatrix} D$ where D is an invertible matrix with real-algebraic entries, and $B \in (\mathbb{A} \cap \mathbb{R})^{2\times 2}$. Indeed, if $\mathcal{A}$ has only real eigenvalues then this is achieved by converting $\mathcal{A}$ to Jordan form, and if $\mathcal{A}$ has complex eigenvalues α and $\overline{\alpha}$, then this is achieved by setting $D = (v, u, w)$ where v is an eigenvector corresponding to 0, and $u + iw$ is an eigenvector corresponding to α. In addition, B is invertible, since its eigenvalues are the nonzero eigenvalues of $\mathcal{A}$.

In the full version, we show that in this case, there exist polytopes $P', R' \subseteq \mathbb{R}^2$ such that for every $n \geq 2$ the following holds: there exists $x \in P$ such that $\mathcal{A}^n x \in R$ iff there exists $x' \in P'$ such that $B^{n-1} x' \in R'$. Thus, it is enough to consider the polytopes P', R' and the invertible matrix B. Moreover, we show that computing P' and R' can be done in polynomial time. We also show a similar approach can be taken when 0 has multiplicity 2 or 3 (with the latter being trivial, since $\mathcal{A}$ is then nilpotent).

It should be noted that in the reduction above, even if the input had only rational entries, the output may still require a real-algebraic description. However, the degree and height of the algebraic numbers involved in the description of the output polytopes remain polynomial in the size of the input.

Finally, we note that we can always *increase* the dimension of the problem while maintaining an invertible matrix. Indeed, Given a invertible matrix $B \in (\mathbb{A} \cap \mathbb{R})^{2\times 2}$, we can consider the invertible matrix $\begin{pmatrix} 1 & 0 \\ 0 & B \end{pmatrix}$, and change $P, R \subseteq \mathbb{R}^2$ to $\{1\} \times P, \{1\} \times R \subseteq \mathbb{R}^3$ (and a similar approach when $B \in (\mathbb{A} \cap \mathbb{R})^{1\times 1}$). Thus, it is enough to solve the problem in the invertible case in dimension 3.

3.3 From the invertible case to an equation system

In this section we focus on solving 3DPCP in the invertible case.

Let P_1, P_2 be the input polytopes (whose description may contain algebraic numbers, as per the reduction of Section 3.2), and let $\mathcal{A} \in (\mathbb{A} \cap \mathbb{R})^{3\times 3}$ be an invertible matrix. By Section 3.1, and since $\mathcal{A}$ is invertible, it suffices to decide whether there exists a number $n \in \mathbb{N}$ such that either there exists a vertex x of P_1 with $\mathcal{A}^n x \in P_2$, or there exists an edge e of P_1 such that $\mathcal{A}^n e$ intersects a face of P_2. Note that we may need to reverse the roles of P_1 and P_2, and use $\mathcal{A}^{-1}$ instead of $\mathcal{A}$. We remark that $\left\|\mathcal{A}^{-1}\right\|$ is polynomial in $\|\mathcal{A}\|$, and moreover – since the eigenvalues of $\mathcal{A}^{-1}$ are inverses of those of $\mathcal{A}$ – the description length of the eigenvalues of $\mathcal{A}^{-1}$ is equal to that of $\mathcal{A}$.

In [8], the authors show that the problem of deciding, given a polyhedron P in $\mathbb{R}^3$, a vector $x \in \mathbb{R}^3$, and a matrix $\mathcal{A} \in (\mathbb{A} \cap \mathbb{R})^{3\times 3}$, whether there exists $n \in \mathbb{N}$ such that $\mathcal{A}^n x \in P$ is solvable in **PSPACE**. This solves the former case. It remains to solve the latter.

We thus assume that we are given as input a matrix $\mathcal{A} \in (\mathbb{A} \cap \mathbb{R})^{3\times 3}$, an edge $E = \{u + \lambda v : \lambda \in J\}$ where $u, v \in \mathbb{R}^3$ and J is either $[0, 1]$ or $[0, \infty)$, and a face $F = \{s + \mu t + \nu r : T(\mu, \nu)\}$, where $s, t, r \in \mathbb{R}^3$ and $T(\mu, \nu)$ is one of the following predicates (as per Lemma 3):

- $T(\mu, \nu) \equiv \mu \geq 0 \wedge \nu \geq 0$,
- $T(\mu, \nu) \equiv \mu \geq 0 \wedge \nu \geq 0 \wedge \mu + \nu \leq 1$,
- $T(\mu, \nu) \equiv \mu \geq 0 \wedge \nu \geq 0 \wedge \nu \leq 1$.

We wish to determine whether there exists a number n and $x \in E$ such that $\mathcal{A}^n x \in F$. In the following, we will treat the case where $E = \{u + \lambda v : \lambda \in [0, 1]\}$ and $F = \{s + \mu t + \nu r : \mu \geq 0 \wedge \nu \geq 0 \wedge \mu + \nu \leq 1\}$. The other cases are slightly simpler, and can be solved *mutatis-mutandis*.

Consider the eigenvalues of $\mathcal{A}$. Since $\mathcal{A}$ is a 3×3 invertible matrix, either all the eigenvalues are real, or there is one real eigenvalue ρ, and two complex, conjugate eigenvalues, α and $\overline{\alpha}$. In the latter case, $\mathcal{A}$ is also diagonalizable. We consider here the latter case. In the full version we show how to handle the former case, which is easier.

Thus, let us assume that the eigenvalues of $\mathcal{A}$ are $\rho \in \mathbb{A}\cap\mathbb{R}$ and $\alpha, \overline{\alpha} \in \mathbb{A}$. We can compute an invertible matrix $B \in \mathbb{A}^{3\times 3}$ such that $A = B^{-1} \begin{pmatrix} \rho & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \overline{\alpha} \end{pmatrix} B$, and the rows of B are the respective eigenvectors. Note that if w_α is an eigenvector of α, then $\overline{w_\alpha}$ is eigenvector of $\overline{\alpha}$, so we can write $B = \begin{pmatrix} w_\rho & w_\alpha & \overline{w_\alpha} \end{pmatrix}^T$. We now have that $\mathcal{A}^n = B^{-1} \begin{pmatrix} \rho^n & 0 & 0 \\ 0 & \alpha^n & 0 \\ 0 & 0 & \overline{\alpha}^n \end{pmatrix} B$ for every $n \in \mathbb{N}$. By analyzing the structure of B and B^{-1}, it is not hard to verify that every entry of $\mathcal{A}^n$ is a linear combination of $\alpha^n, \overline{\alpha}^n$ and ρ^n such that the coefficients of α^n and $\overline{\alpha}^n$ are conjugates, and the coefficient of ρ^n is real. That is, for every $1 \leq i, j \leq 3$ it holds that $(\mathcal{A}^n)_{i,j} = c_{i,j}\alpha^n + \overline{c_{i,j}}\,\overline{\alpha}^n + d_{i,j}\rho^n$ for coefficients $c_{i,j} \in \mathbb{A}$ and $d_{i,j} \in \mathbb{A} \cap \mathbb{R}$ (independent of n).

Consider a vector $x = u + \lambda v \in E$. We can write $\mathcal{A}^n x = \mathcal{A}^n u + \lambda \mathcal{A}^n v$, and observe that for $1 \leq i \leq 3$ we have $(\mathcal{A}^n u)_i = (c_{i,1}u_1 + c_{i,2}u_2 + c_{i,3}u_3)\alpha^n + \left(\overline{c_{i,1}u_1 + c_{i,2}u_2 + c_{i,3}u_3}\right)\overline{\alpha}^n + (d_{i,1}u_1 + d_{i,2}u_2 + d_{i,3}u_3)\rho^n$, and a similar structure holds for $\mathcal{A}^n v$. By renaming the coefficients, we can write $(\mathcal{A}^n u + \lambda \mathcal{A}^n v)_i = f_i\alpha^n + \overline{f_i}\,\overline{\alpha}^n + g_i\rho^n + \lambda(h_i\alpha^n + \overline{h_i}\,\overline{\alpha}^n + k_i\rho^n)$ where $f_i, h_i \in \mathbb{A}$ and $g_i, k_i \in \mathbb{A} \cap \mathbb{R}$ for $1 \leq i \leq 3$.

We can now formulate the problem as follows: does there exist a number $n \in \mathbb{N}$ such that the following first-order sentence is true: $\exists \lambda, \mu, \nu : 0 \leq \lambda, \mu, \nu \leq 1 \wedge \mu + \nu \leq 1 \wedge$

$$\bigwedge_{i=1}^{3} \left(f_i\alpha^n + \overline{f_i}\,\overline{\alpha}^n + g_i\rho^n + \lambda(h_i\alpha^n + \overline{h_i}\,\overline{\alpha}^n + k_i\rho^n) = s_i + \mu t_i + \nu r_i\right) . \tag{3}$$

As mentioned in Section 2.2, we can convert (3) to an equivalent, quantifier-free sentence. Since our reasoning requires this equivalent sentence to have a special structure, we must explicitly remove the quantifiers. This is done in the full version using Fourier-Motzkin quantifier elimination [11], where we conclude the following.

▶ **Theorem 4.** *There exist constants M, M' such that the sentence (3) is equivalent to a disjunction $\bigvee_{i=1}^{M} \mathrm{Sys}_i$ where for every $1 \leq i \leq M$, Sys_i is a conjunction of at most M' expressions of the form*

$$A\alpha^{2n} + \overline{A}\overline{\alpha}^{2n} + B\alpha^n\rho^n + \overline{B}\overline{\alpha}^n\rho^n + C\rho^{2n} + D|\alpha|^{2n} + E\alpha^n + \overline{E}\overline{\alpha}^n + F\rho^n + G \bowtie 0\,, \quad (4)$$

where $\bowtie \in \{>, =\}$, $A, B, E \in \mathbb{A}$, and $C, D, F, G \in \mathbb{A} \cap \mathbb{R}$. Moreover, the description of Sys *is polynomial in $\|I\|$ (the description length of the input).*

4 Solving the System

This section constitutes the main technical challenge of the paper, namely to decide whether there exists $n \in \mathbb{N}$ such that the disjunction presented in Theorem 4 is true. We refer to such an n as a *solution* for the disjunction.

We first note that it is enough to consider each system in the disjunction separately. Indeed, since the number of systems is bounded, independent of the input, we can try to solve each one separately. Our goal is then to decide, given a system Sys of expressions as per Theorem 4, whether there exists a solution $n \in \mathbb{N}$ that satisfies all the expressions simultaneously.

We divide our analysis to two cases. First we handle the (straightforward) case where $\frac{\alpha}{|\alpha|}$ is a root of unity. We then proceed to consider the more involved case, where $\frac{\alpha}{|\alpha|}$ is not a root of unity.

4.1 The case where $\frac{\alpha}{|\alpha|}$ is a root of unity

Suppose that $\frac{\alpha}{|\alpha|}$, denoted γ, is a root of unity. We can now treat (4) as

$$|\alpha|^{2n}A\gamma^{2n} + |\alpha|^{2n}\overline{A}\overline{\gamma}^{2n} + |\alpha|^n B\gamma^n\rho^n + |\alpha|^n\overline{B}\overline{\gamma}^n\rho^n + C\rho^{2n} + D|\alpha|^{2n} + |\alpha|^n E\gamma^n + |\alpha|^n\overline{E}\overline{\gamma}^n + F\rho^n + G \bowtie 0\,.$$

Let d be the order of γ, then γ^2 is also a root of unity of order at most d. Thus, there are at most d^2 possible values for (γ^n, γ^{2n}), determined by the pair $(n \bmod d, 2n \bmod d)$. We can now treat the expression as d^2 expressions of real-algebraic sums of exponentials. We show that $d \leq \deg(\gamma)^2$, so these can be solved in **PSPACE** using standard techniques of asymptotic analysis, by considering the coefficients and the moduli of α and ρ (see the full version for details).

4.2 The case where $\frac{\alpha}{|\alpha|}$ is not a root of unity

When $\gamma = \frac{\alpha}{|\alpha|}$ is not a root of unity, things are more involved. Nonetheless, we prove the following theorem.

▶ **Theorem 5.** *The problem of deciding whether a system* Sys *of expressions of the form* (4) *has a solution, is in **PSPACE**.*

Before proving the theorem, we need some definitions. In the following, we assume w.l.o.g. that $\rho > 0$. Indeed, if $\rho < 0$ then we can divide into two cases according to the parity of n, and solve each separately (note that $\rho \neq 0$ since the matrix $\mathcal{A}$ is invertible).

For an expression of the form (4), we obtain its *normalized expression* by dividing it by $(\max\{|\alpha|^2, |\alpha|\rho, \rho^2, |\alpha|, |\rho|\})^n$ (and such that the coefficient of the element we divide by is nonzero). Thus, the normalized expression is of the form

$$A\gamma^{2n} + \overline{A}\overline{\gamma}^{2n} + B\gamma^n + \overline{B}\overline{\gamma}^n + C + r(n) \bowtie 0, \tag{5}$$

with $\gamma \in \mathbb{A}$ such that $|\gamma| = 1$ and γ is not a root of unity, $A, B \in \mathbb{A}$ and $C \in \mathbb{A} \cap \mathbb{R}$ are not all 0, and $r(n) = \sum_{l=1}^{m} D_l \beta_l^n + \overline{D_l \beta_l}^n$, where $|\beta_l| < 1$ for every $1 \leq l \leq m$, and $0 \leq m \leq 4$ (note that for uniformity we treat real numbers in $r(n)$ as a sum of complex conjugates). For every $1 \leq l \leq m$, β_l is a quotient of two elements from the set $\{\alpha, \alpha^2, \rho, \rho^2, \alpha\rho\}$. Since α and ρ are eigenvalues of $\mathcal{A}$, $\deg(\alpha), \deg(\rho)$ are $\|\mathcal{A}\|^{O(1)}$. Thus, by Section 2.1, $\deg(\beta_l) = \|\mathcal{A}\|^{O(1)}$, and $H(\beta_l) = 2^{\|\mathcal{A}\|^{O(1)}}$.

Since γ is not a root of unity, then $\{\gamma^n : n \in \mathbb{N}\}$ is dense in the unit circle. With this motivation in mind, we define, for a normalized expression, its *dominant function* $f : \mathbb{C} \to \mathbb{R}$ as $f(z) = Az^2 + \overline{A}\overline{z}^2 + Bz + \overline{B}\overline{z} + C$. Observe that (5) is now equivalent to $f(\gamma^n) + r(n) \bowtie 0$.

The following lemma is our main technical tool in proving Theorem 5.

▶ **Lemma 6.** *Consider a normalized expression as in* (5)*. Let $\|I\|$ be its encoding length, and let f be its dominant function. Then there exists $N \in \mathbb{N}$ computable in polynomial time in $\|I\|$ with $N = 2^{\|I\|^{O(1)}}$ such that for every $n > N$ it holds that*

1. $f(\gamma^n) \neq 0$,
2. $f(\gamma^n) > 0$ *iff* $f(\gamma^n) + r(n) > 0$,
3. $f(\gamma^n) < 0$ *iff* $f(\gamma^n) + r(n) < 0$.

In particular, the lemma implies that if $f(n) + r(n) = 0$, then $n \leq N$. The proof of Lemma 6 relies on the following lemma from [18], which is itself a consequence of the Baker-Wüstholz Theorem [2].

▶ **Lemma 7** ([18])**.** *There exists $D \in \mathbb{N}$ such that for all algebraic numbers ζ, ξ of modulus 1, and for every $n \geq 2$, if $\zeta^n \neq \xi$, then $|\zeta^n - \xi| > \frac{1}{n^{(\|\zeta\| + \|\xi\|)^D}}$.*

We now turn to prove Lemma 6. The following synopsis contains the main ideas. The full proof can be found in the full version.

Proof (Synopsis). Since $\{\gamma^n : n \in \mathbb{N}\}$ is dense on the unit circle, we consider $f(z)$ for z in the unit circle. In the full proof, we show that $\{z : f(z) = 0 \wedge |z| = 1\}$ contains at most four points $\{z_1, \ldots, z_4\}$, whose coordinates are algebraic. Since γ is not a root of unity, it holds that $\gamma^{n_1} \neq \gamma^{n_2}$ for every $n_1 \neq n_2 \in \mathbb{N}$. Thus, there exists $N_1 \in \mathbb{N}$ such that $\gamma^n \notin \{z_1, \ldots, z_4\}$ for every $n > N_1$. Moreover, by Lemma D.1 in [6], we have that $N_1 = k^{O(1)}$, where $k = \|\gamma\| + \sum_{j=1}^{4} \|z_j\|$, and N_1 can be computed in polynomial time in k. Then, by Lemma 7, there exists a constant $D \in \mathbb{N}$ such that for every $n \geq N_1$ and $1 \leq j \leq 4$ we have that $|\gamma^n - z_j| > \frac{1}{n^{(k^D)}}$. Intuitively, for $n > N_1$ we have that γ^n does not get close to any z_i "too quickly" as a function of n. In particular, for $n > N_1$ we have $f(\gamma^n) \neq 0$. It thus remains to show that for large enough n, $r(n)$ does not affect the sign of $f(\gamma^n) + r(n)$. Intuitively, this is the case because $r(n)$ decreases exponentially, while $|f(\gamma^n)|$ is bounded from below by an inverse polynomial. While proving that this holds in general is not very difficult, note that we also need the bound on N in the statement of the Lemma to be effectively computable and to be $2^{\|I\|^{O(1)}}$, which complicates things significantly.

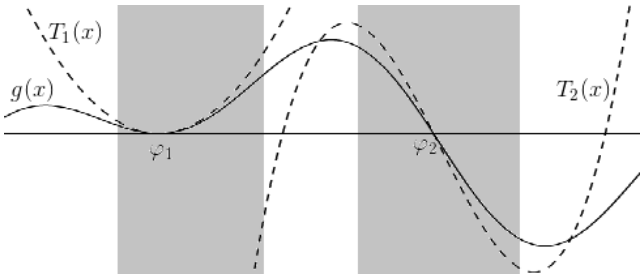


Figure 1 $g(x)$ and two Taylor polynomials: $T_1(x)$ around φ_1 and $T_2(x)$ around φ_2. The shaded regions show where requirements (1)–(3) hold, which determine ϵ_1. Observe that for T_1, the most restrictive requirement is $|g(x) - T_1(x)| \leq \frac{1}{2}T_1(x)$, whereas for T_2 the restriction is the requirement that $T_2(x)$ is monotone.

We consider the function $g : (-\pi, \pi] \to \mathbb{R}$ defined by $g(x) = f(e^{ix})$. Explicitly, we have $g(x) = 2|A|\cos(2x + \theta_A) + 2|B|\cos(x + \theta_B) + C$ where $\theta_A = \arg(A)$ and $\theta_B = \arg(B)$. By the above, g has at most four roots, denoted $\varphi_1, \ldots, \varphi_4$. We now show that there exist $N_2 \in \mathbb{N}$ and a non-negative polynomial $p(n)$ such that $f(\gamma^n) = g(\arg(\gamma^n)) > \frac{1}{p(n)}$ for every $n > N_2$. For every $1 \leq j \leq 4$ consider the first non-zero Taylor polynomial T_j of g around φ_j. In the full version we show that the degree of such approximations is at most 3. We show that there exists $\epsilon_1 > 0$ such that for every $x \in (\varphi_j - \epsilon_1, \varphi_j + \epsilon)$ it holds that (1) $|g(x) - T_j(x)| \leq \frac{1}{2}|T_j(x)|$, (2) g is monotone on either side of φ_j, and (3) T is monotone with the same tendency of g (see Figure 1 for an illustration). In the full version we also show that crucially, we can require ϵ_1 to be efficiently computable and $\frac{1}{\epsilon} = 2^{n^{O(1)}}$.

Consider $n \in \mathbb{N}$ such that $\gamma^n \in \bigcup_{j=1}^{4}(\varphi_j - \epsilon_1, \varphi_j + \epsilon_1)$ and such that $n > N_1$, then as we have seen above, $\frac{1}{n^{(k^D)}} < |\gamma^n - z_j|$. But $|\gamma^n - z_j| < |\arg(\gamma^n) - \varphi_j|$ (since the euclidean distance is smaller than the arc length), so $|\arg(\gamma^n) - \varphi_j| > \frac{1}{n^{(k^D)}}$. From requirements (1) and (2) of ϵ_1, we get that $|g(\arg(\gamma^n))| \geq \frac{1}{2}|T_j(\gamma^n)|$ and from the monotonicity of T_j in the neighbourhood of φ_j (requirement (3)), we have that $\frac{1}{2}|T_j(\gamma^n)| > \frac{1}{2}\min\left\{|T_j(\varphi_j + \frac{1}{n^{(k^D)}})|, |T_j(\varphi_j - \frac{1}{n^{(k^D)}})|\right\}$, from which we conclude that $|g(\arg(\gamma^n))| > \frac{1}{p(n)}$ for some non-negative polynomial p. Moreover, we can compute the representation of p in polynomial time.

Finally, for $x \notin \bigcup_{j=1}^{4}(\varphi_j - \epsilon_1, \varphi_j + \epsilon_1)$, we have that $|g(x)|$ is bounded from below by a constant. Our careful accounting of $\|\epsilon_1\|$ in the full version allows us to compute this bound, and show that it is not too small.

The last step in the proof is to show that $r(n)$ decreases fast enough such that $r(n) < \frac{1}{p(n)}$ for every $n > N_3$ for some large enough $N_3 \in \mathbb{N}$. Clearly this holds eventually, since $r(n)$ decreases exponentially. However, we also need a bound on the size of N_3, which requires more effort. Recall that $r(n) = \sum_{l=1}^{m} D_l\beta_l^n + \overline{D_l\beta_l}^n$. By applying The root separation bound (2) from Section 2.1 to $1 - |\beta_l|$, we compute $\epsilon \in (0, 1)$ and $N_3 \in \mathbb{N}$ such that $\frac{1}{\epsilon}$ and N_3 are $2^{\|I\|^{O(1)}}$, and for every $n > N_3$ it holds that $|r(n)| < (1 - \epsilon)^n$. Using this, we can find $N_4 \in \mathbb{N}$ such that $N_4 = 2^{\|I\|^{O(1)}}$ and $|r(n)| < \frac{1}{p(n)}$ for all $n > N_4$, from which we can conclude the proof. ◀

We are now ready to prove Theorem 5

Proof. For every expression in Sys, let f be the corresponding function as per Lemma 6, and compute its respective bound N. If $\bowtie$ is "=", then by Lemma 6, if the equation is satisfiable for $n \in \mathbb{N}$, then $n < N$.

If all the $\bowtie$ are ">", then for each such inequality compute $\{z : f(z) > 0\}$. If the intersection of these sets is empty, then if n is a solution for the system, it must hold that $n < N$. If the intersection is non-empty, then it is an open set. Since γ is not a root of unity, then $\{\gamma^n : n \in \mathbb{N}\}$ is dense in the unit circle. Thus, there exists $n > N$ such that γ^n is in the above intersection, so the system has a solution. Checking the emptiness of the intersection can be done in polynomial time using Theorem 2.

Thus, it remains to check whether there exists a solution $n < N$. Recall that $N = 2^{\|I\|^{O(1)}}$. Thus, in order to check whether the system is solved for $n < N$, we need to compute, e.g., α^{2n}, whose representation is exponential in $\|I\|$, so a naive implementation would take exponential space.

Instead, we take a similar approach to [8]: by representing numbers as arithmetic circuits, deciding the positivity (or testing for 0 equality) can be done using an oracle to **PosSLP**, which by [1] is in the counting hierarchy. By first guessing $n < N$, the problem can be solved in $\mathbf{NP}^{\mathbf{PosSLP}}$, which is contained in **PSPACE**. ◀

5 Conclusions

5.1 Proof of Theorem 1

We conclude by giving an explicit proof of Theorem 1: Given polytopes P_1 and P_2 and a matrix $\mathcal{A}$, if $\mathcal{A}$ is singular, we first apply (in polynomial time) the reduction in Section 3.2. Thus, we can assume $\mathcal{A}$ is invertible. Next, if P_1 or P_2 are unbounded, for each unbounded face F we proceed as follows: decompose F as per Lemma 3, so $F = \bigcup_{i=1}^{m} A_i$, and recall that iterating over the A_i's can be done in **PSPACE**. In each iteration, consider an edge E of P_1 and a face F of P_2 (both of which may belong to sets A_i as above). Formulate the first-order sentence (3) in Section 3.3, and apply Theorem 4 to obtain an equivalent disjunction of systems $\bigvee_{i=1}^{M} \mathrm{Sys_i}$, where M is constant. Then, for each system $\mathrm{Sys_i}$, check in **PSPACE** whether it has a solution, using either Section 4.1 or Theorem 5. If no solution was found, check in **PSPACE** whether a vertex of P_1 collides with P_2, using the algorithm in [8]. Then, if still no solution is found, repeat the same procedure by interchanging the roles of P_1 and P_2, and considering the matrix $\mathcal{A}^{-1}$ instead of $\mathcal{A}$. The correctness and complexity of this procedure follow from the proofs of the respective theorems.

5.2 Discussion

This paper studies an extension of the Orbit Problem, in which the input is existentially quantified over a polytope, and the target is a polytope. The importance of this work is twofold: from a practical perspective, we provide an algorithm for deciding the termination of linear while loops with affine guards, up to dimension three, when the input is not fixed. From a more theoretical perspective, and as already pointed out by Kannan and Lipton in [14], the Orbit Problem and its variants are closely related to long-standing open problems such as the *Skolem Problem*, and various number-theoretic problems. It is therefore useful and compelling to push the borders of decidability, in order to identify the core of the remaining difficulties, and to eventually hopefully overcome them.

Finally, as discussed in Section 1, the problem at hand can be viewed as a particular case of the Orbit Problem in dimension six where the target is a semi-algebraic set. As the general problem is known to be hard even in dimension four, our work here suggests that interesting and useful fragments are tractable even in high dimensions.

References

1 Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009.

2 Alan Baker and Gisbert Wüstholz. Logarithmic forms and group varieties. *J. reine angew. Math.*, 442(19-62):3, 1993.

3 Saugata Basu, Richard Pollack, and Marie-Francoise Roy. *Algorithms in real algebraic geometry*, volume 20033. Springer, 2005.

4 Amir M. Ben-Amram and Samir Genaim. Ranking functions for linear-constraint loops. *Journal of the ACM (JACM)*, 61(4):26, 2014.

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

6 Ventsislav Chonev, Joël Ouaknine, and James Worrell. On the complexity of the orbit problem. *arXiv preprint arXiv:1303.2981*, 2013.

7 Ventsislav Chonev, Joël Ouaknine, and James Worrell. The orbit problem in higher dimensions. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 941–950. ACM, 2013.

8 Ventsislav Chonev, Joël Ouaknine, and James Worrell. The polyhedron-hitting problem. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 940–956. SIAM, 2015.

9 Henri Cohen. *A course in computational algebraic number theory*, volume 138. Springer Science & Business Media, 2013.

10 George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975*, pages 134–183. Springer, 1975.

11 Jean Baptiste Joseph Fourier. Solution d'une question particuliere du calcul des inégalités. *Nouveau Bulletin des Sciences par la Société philomatique de Paris*, 99:100, 1826.

12 Michael A. Harrison. Lectures on linear sequential machines. Technical report, DTIC Document, 1969.

13 Ravindran Kannan and Richard J. Lipton. The orbit problem is decidable. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 252–261. ACM, 1980.

14 Ravindran Kannan and Richard J. Lipton. Polynomial-time algorithm for the orbit problem. *Journal of the ACM (JACM)*, 33(4):808–821, 1986.

15 David Lee and Mihalis Yannakakis. Online minimization of transition systems. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 264–274. ACM, 1992.

16 Maurice Mignotte. Some useful bounds. In *Computer algebra*, pages 259–263. Springer, 1983.

17 Joël Ouaknine, João Sousa Pinto, and James Worrell. On termination of integer linear loops. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 957–969. SIAM, 2015.

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

19 Victor Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers & Mathematics with Applications*, 31(12):97–138, 1996.

20 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. part i: Introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *Journal of Symbolic Computation*, 13(3):255–299, 1992.

21 Terence Tao. *Structure and randomness: pages from year one of a mathematical blog.* American Mathematical Soc., 2008.

22 Alfred Tarski. A decision method for elementary algebra and geometry. *Bulletin of the American Mathematical Society*, 59, 1951.

Dynamic Time Warping and Geometric Edit Distance: Breaking the Quadratic Barrier*

Omer Gold[1] **and Micha Sharir**[2]

1 **Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
omergold@post.tau.ac.il

2 **Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
michas@post.tau.ac.il

Abstract

Dynamic Time Warping (DTW) and Geometric Edit Distance (GED) are basic similarity measures between curves or general temporal sequences (e.g., time series) that are represented as sequences of points in some metric space (X, dist). The DTW and GED measures are massively used in various fields of computer science and computational biology, consequently, the tasks of computing these measures are among the core problems in P. Despite extensive efforts to find more efficient algorithms, the best-known algorithms for computing the DTW or GED between two sequences of points in $X = \mathbb{R}^d$ are long-standing dynamic programming algorithms that require quadratic runtime, even for the one-dimensional case $d = 1$, which is perhaps one of the most used in practice.

In this paper, we break the nearly 50 years old quadratic time bound for computing DTW or GED between two sequences of n points in $\mathbb{R}$, by presenting deterministic algorithms that run in $O\left(n^2 \log\log\log n / \log\log n\right)$ time. Our algorithms can be extended to work also for higher dimensional spaces $\mathbb{R}^d$, for any constant d, when the underlying distance-metric dist is polyhedral (e.g., L_1, L_∞).

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations

Keywords and phrases Dynamic Time Warping, Geometric Edit Distance, Time Series, Point Matching, Geometric Matching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.25

1 Introduction

Searching for optimal algorithms is a standard routine in the study of algorithm design. Among the most popular basic problems in P are those that have standard algorithms that run in $O(n^c)$ time, where $c = 2$ or 3. For $c = 3$ (cubic time), we can find many kinds of combinatorial matrix multiplication algorithms, and for $c = 2$ (quadratic time), we can find many fundamental problems, such as 3SUM, and many basic matching problems between strings, curves, and point-sequences, such as Edit Distance, Geometric Edit Distance (GED), Dynamic Time Warping (DTW), Discrete Fréchet Distance, and Longest Common Subsequence (LCS). These problems are usually referred to as "quadratic problems".

* Work on this paper has been supported by Grant 892/13 from the Israel Science Foundation, by Grant 2012/229 from the U.S.-Israeli Binational Science Foundation, by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 25; pp. 25:1–25:14

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

Motivated to find optimal algorithms for these basic problems, researchers have come up with time bounds of the form $O(n^c/\operatorname{polylog}(n))$, where $\operatorname{polylog}(n)$ stands for $\log^k n$, for some constant $k > 0$. By now, many classical quadratic problems have upper bounds of the form $O(n^2/\operatorname{polylog}(n))$, including all of the problems mentioned above, except for DTW and GED; see [3, 29, 18, 21] for such upper bounds. Among the very few archetypal quadratic problems for which no $o(n^2)$-time algorithm is known, DTW and GED seem to be prominent examples, considering the decades of extensive efforts to break the quadratic barrier.

Motivation. Complementary to the standard theoretical interest in finding optimal algorithms for basic problems in P, a significant progress has been made in recent years towards a better understanding these problems, by proving conditional lower bounds via reductions from basic problems, such as 3SUM and CNF-SAT. Assuming that CNF-SAT takes $\Omega\left(2^{(1-o(1))n}\right)$ time (the so-called *Strong Exponential Time Hypothesis* (SETH) [22, 23]), has led to recent lower bounds for a growing list of problems, including most of the quadratic problems mentioned above. Specifically, assuming SETH, there is no $O\left(n^{2-\Omega(1)}\right)$-time algorithm for Discrete Fréchet Distance [8], Edit Distance [6], LCS [1, 9], and DTW [1, 9].

A recent seminal work by Abboud *et al.* [2] shows that even an improvement by a sufficient polylogarithmic factor for any of these basic problems would lead to major consequences, such as faster Formula-SAT algorithms, and new circuit lower bounds. Hence, the work of Abboud *et al.* highly motivates and revives the study of polylogarithmic-factor improvements for these basic problems, since it may be the only way to push the efficiency of the solution "to the limit".

Problem Statement. Let $A = (p_1, \ldots, p_n)$ and $B = (q_1, \ldots, q_m)$ be two sequences of points (also referred to as curves) in some metric space (X, dist). A *coupling* $C = (c_1, \ldots, c_k)$ between A and B is an ordered sequence of distinct pairs of points from $A \times B$, such that $c_1 = (p_1, q_1)$, $c_k = (p_n, q_m)$, and

$$c_r = (p_i, q_j) \Rightarrow c_{r+1} \in \big\{(p_{i+1}, q_j),\ (p_i, q_{j+1}),\ (p_{i+1}, q_{j+1})\big\},$$

for $r < k$. The DTW-distance between A and B is

$$\mathsf{dtw}(A, B) = \min_{C:\, coupling} \sum_{(p_i, q_j) \in C} \mathsf{dist}(p_i,\, q_j). \tag{1}$$

The coupling C for which the above sum is minimized is called the *optimal coupling*. The DTW problem is to compute $\mathsf{dtw}(A, B)$, and sometimes also the optimal coupling C.

A *monotone matching* $\mathcal{M} = \{m_1, \ldots, m_k\}$ between A and B is a set of pairs of points from $A \times B$, such that any two pairs (p_i, q_j), $(p_{i'}, q_{j'}) \in \mathcal{M}$ satisfy that $i \leq i'$ iff $j \leq j'$, and each point in A is matched with at most one point in B and vice versa (possibly some points in $A \cup B$ do not appear in any pair of the matching); Note the difference from coupling (defined above), which covers all points of $A \cup B$ and a point can appear in multiple pairs of the coupling. The cost of $\mathcal{M}$ is defined to be the sum of all the distances between the points of each pair in $\mathcal{M}$, plus a *gap* penalty parameter $\rho \in \mathbb{R}$, for each point in $A \cup B$ that does not appear in any pair of $\mathcal{M}$.

The Geometric Edit Distance (GED) between A and B is

$$\mathsf{ed}(A, B) = \min_{\mathcal{M}} \sum_{(p_i, q_j) \in \mathcal{M}} \mathsf{dist}(p_i,\, q_j) + \rho\left(n + m - 2|\mathcal{M}|\right), \tag{2}$$

where the minimum is taken over all sets $\mathcal{M}$ of monotone matchings in the complete bipartite graph $A \times B$. The monotone matching $\mathcal{M}$ for which the above sum is minimized is called

the *optimal matching*. The GED problem is to compute $\mathsf{ed}(A, B)$, and sometimes also the optimal matching. More sophisticated gap penalty functions have been proposed [14], but for this presentation, we focus on the standard linear gap penalty function, although our presented algorithm supports more complex gap penalty, such as taking ρ to be a linear function in the coordinates of the points $A \cup B$. By tuning ρ correctly, meaningful matchings can be computed even when faced with outlier points that arise from measurement errors or short deviations in otherwise similar trajectories.

The DTW-distance and GED are massively used in dozens of applications, such as speech recognition, geometric shape matching, DNA and protein sequences, protein backbones, matching of time series data, GPS, video and touch screen authentication trajectories, music signals, and countless data mining applications; see [11, 13, 15, 28, 27, 25, 30, 32, 26] for some examples.

The best-known worst-case running times for solving DTW or GED are given by long-standing dynamic programming algorithms that require $\Theta(nm)$ time. We review the standard quadratic-time DTW and GED algorithms in the full version of this paper [19].

DTW was perhaps first introduced as a speech discrimination method [33] back in the 1960's. GED is a natural extension of the well-known string version of Edit Distance, however, the subquadratic-time algorithms for the string version do not seem to extend to GED (see below).

A popular setting in both theory and practice is the one-dimensional case $X = \mathbb{R}$ (under the standard distance $\mathsf{dist}(x, y) = |x - y|$). Even for this special case, no subquadratic-time algorithms have been known. We mainly consider this case throughout the paper.

Prior Results. Since no subquadratic-time algorithm is known for computing DTW, a number of heuristics were designed to speed up its exact computation in practice; see Wang *et al.* [34] for a survey. Very recently, Agarwal *et al.* [4] gave a near-linear approximation scheme for computing DTW or GED for a restricted, although quite large, family of curves.

Recently, Bringmann and Künnemann [9] proved that DTW on one-dimensional point sequences whose elements are taken from $\{0, 1, 2, 4, 8\} \subset \mathbb{R}$ has no $O(n^{2-\Omega(1)})$-time algorithm, unless SETH fails. They proved a similar hardness result also for Edit Distance between two binary strings, improving the conditional lower bound of Backurs and Indyk [6]. This line of work was extended in a very recent work by Abboud *et al.* [2], mentioned above, where they show that even a sufficiently large polylog(n)-factor improvement over the quadratic time upper bound for Edit Distance or DTW, will lead to major consequences.

Masek and Paterson [29] showed that Edit Distance between two strings of length at most n over an $O(1)$-size alphabet can be solved in $O(n^2/\log n)$ time. More recent works [7, 20] managed to lift the demand for $O(1)$-size alphabet and retain a subquadratic-time bound by making a better use of the word-RAM model. However, these works do not seem to extend to GED, especially not when taking sequences of points with arbitrary real coordinates. In the string version, the cost of replacing a character is fixed (usually 1), hence, we only need to detect that two characters are not identical in order to compute the replacement cost, unlike in GED, where the analogous cost for two matched points is taken to be their distance, under some metric.

Our Results and Related Work. Efforts for breaking the quadratic time barrier for basic similarity measures between curves and point-sequences were recently stimulated by the result of Agarwal *et al.* [3] who showed that the discrete Fréchet distance can be computed in $O(n^2/\log n)$ time. Their algorithm for (discrete) Fréchet distance does not extend to DTW

or GED, as the recursive formula for the (discrete) Fréchet distance uses the max function, while the formula for DTW and GED involves the sum. As a result, the Fréchet distance is effectively determined by a single pair of sequence elements, which fits well into the use of the Four-Russians technique [5], while the DTW and GED are determined by many pairs of elements. This makes our algorithms much more subtle, involving a combination and extension of techniques from computational geometry and graph shortest paths.

To simplify the presentation, we present our results only for the "balanced" case $m = n$; extending them to the general case $m \leq n$ is easy. The standard $\Theta(mn)$-time algorithm is superior only when m is subpolynomial in n.

▶ **Theorem 1.** *Given two sequences $A = (p_1, \ldots, p_n)$ and $B = (q_1, \ldots, q_n)$, each of n points in $\mathbb{R}$, the* DTW*-distance* $\mathsf{dtw}(A, B)$ *(and optimal coupling), or the* GED $\mathsf{ed}(A, B)$ *(and optimal matching) can be computed by a deterministic algorithm in $O(n^2 \log\log\log n / \log\log n)$ time.*

Theorem 1 gives the very first subquadratic-time algorithm for solving DTW, breaking the nearly 50 years old $\Theta(n^2)$ time bound. In the full version of this paper [19], we extend our algorithm to give a more general result, which supports high-dimensional polyhedral metric spaces, as stated in Theorem 2. Additionally, in [19], we extend our algorithm for solving GED.

▶ **Theorem 2.** *Let $A = (p_1, \ldots, p_n)$ and $B = (q_1, \ldots, q_n)$ be two sequences of n points in a polyhedral metric space*[1] *($\mathbb{R}^d$, dist). Then* $\mathsf{dtw}(A, B)$ *(and optimal coupling), or* $\mathsf{ed}(A, B)$ *(and optimal matching) can be computed by a deterministic algorithm in $O(n^2 \log\log\log n / \log\log n)$ time, for any constant d.*

2 Preliminaries

Throughout the paper, we view matrices with rows indexed in increasing order from bottom to top and columns indexed in increasing order from left to right, so, for example, $M[1, 1]$ the leftmost-bottom cell of a matrix M.

In Fredman's classic 1976 articles on the decision tree complexity of $(\min, +)$-matrix multiplication [17], and on sorting $X + Y$ [16], he often uses the simple observation that $a + b < a' + b'$ iff $a - a' < b' - b$. This observation is usually referred to as *Fredman's trick*. In our algorithm, we will often use the following extension of Fredman's trick.

$$\begin{aligned} a_1 - b_1 + \cdots + a_r - b_r < a'_1 - b'_1 + \cdots + a'_t - b'_t \\ \textit{if and only if} \\ a_1 + \cdots + a_r - a'_1 - \cdots - a'_t < b_1 + \cdots + b_r - b'_1 - \cdots - b'_t. \end{aligned} \tag{3}$$

Our algorithm uses the following geometric domination technique, based on an algorithm by Chan [12]. Given a finite set Q of red points and blue points in $\mathbb{R}^d$, the *bichromatic dominating pairs reporting* problem is to report all the pairs $(p, q) \in Q^2$ such that p is red, q is blue, and p dominates q, i.e., p is greater than or equal to q at each of the d coordinates. A natural divide-and-conquer algorithm [31, p. 366] runs in $O(|Q| \log^d |Q| + K)$ time, where K is the output size. Chan [12] provided an improved *strongly* subquadratic time bound (excluding the cost of reporting the output) when $d = O(\log |Q|)$, with a sufficiently small constant of proportionality.

[1] That is, the underlying metric is induced by a norm, whose unit ball is a symmetric convex polytope with $O(1)$ facets (e.g., L_1, L_∞).

▶ **Lemma 3** (Chan [12]). *Given a finite set $Q \subset \mathbb{R}^d$ of red and blue points, one can compute all bichromatic dominating pairs $(p, q) \in Q^2$ in time $O(c_\varepsilon^d |Q|^{1+\varepsilon} + K)$, where K is the output size, $\varepsilon \in (0, 1)$ is an arbitrary prespecified parameter, and $c_\varepsilon = 2^\varepsilon/(2^\varepsilon - 1)$.*

Throughout the paper, we invoke Lemma 3 many times, with $\varepsilon = 1/2, c_\varepsilon \approx 3.42$, and $d = \delta \log n$, where $\delta > 0$ is a sufficiently small constant, chosen to make the overall running time of all the invocations dominated by the total output size; see below for details.

We denote by $[N] = \{1, \ldots, \lceil N \rceil\}$, the set of the first $\lceil N \rceil$ natural numbers, for any $N \in \mathbb{R}^+$.

Throughout the paper, we sometimes refer to a square matrix as a *box*.

Our model of computation is a simplified Real RAM model, in which "truly real" numbers are subject to only two unit-time operations: addition and comparison. In all other respects, the machine behaves like a $w = O(\log n)$-bit word RAM with the standard repertoire of unit-time AC^0 operations, such as bitwise Boolean operations, and left and right shifts.

3 Dynamic Time Warping in Subquadratic Time

As above, the input consists of two sequences $A = (p_1, \ldots, p_n)$ and $B = (q_1, \ldots, q_n)$ of n points in $\mathbb{R}$. Our algorithm can easily be modified to support the case where A and B have different lengths.

Preparations. We fix some (small) parameter g, whose value will be specified later; for simplicity, we assume that $\frac{n}{g-1}$ is an integer. We decompose A and B into $s = \frac{n}{g-1}$ subsequences $A_1, \ldots, A_s$, and $B_1, \ldots, B_s$, such that for each $i, j \in \{2, \ldots, s\}$, each of A_i and B_j consists of $g - 1$ consecutive elements of the corresponding sequence, prefixed by the last element of the preceding subsequence. We have that A_1 and B_1 are both of size $g - 1$, each A_i and B_j is of size g, for each $i, j \in \{2, \ldots, s\}$, and each consecutive pairs A_i, A_{i+1} or B_j, B_{j+1} have one common element.

For each $i, j \in [s]$, denote by $D_{i,j}$ the *all-pairs-distances matrix* between points from A_i and points from B_j; specifically, $D_{i,j}$ is a $g \times g$ matrix (aka *box*) (see below for the cases $i = 1$ or $j = 1$) such that for every $\ell, m \in [g]$,

$$D_{i,j}[\ell, m] = \big|A_i(\ell) - B_j(m)\big|.$$

For all $i \in [s]$, we add a leftmost column with ∞ values to each box $D_{i,1}$, and similarly, we add a bottommost row with ∞ values to each box $D_{1,i}$. In particular, $D_{1,1}$ is augmented by both new leftmost column and new bottommost row. The common element $D_{1,1}[0, 0]$ of this row and column is set to 0. Overall, we have $s^2 = \left(\frac{n}{g-1}\right)^2$ boxes $D_{i,j}$, all of size $g \times g$.

We define a *staircase path* P on a $g \times g$ matrix $D_{i,j}$ as a sequence of positions from $[g] \times [g]$ that form a monotone staircase structure, starting from a cell on the left or bottom boundary and ending at the right or top boundary, so that each subsequent position is immediately either to the right, above, or above-right of the previous one. Formally, by enumerating the path positions as $P(0), \ldots, P(t^*)$, we have $P(t+1) \in \{P(t)+(0,1), P(t)+(1,0), P(t)+(1,1)\}$, for each $t = 0, \ldots, t^* - 1$. The path starts at some point $P(0) = (\cdot, 1)$ or $(1, \cdot)$, which lies on either the left or the bottom boundary, and ends at some t^* (not necessarily the first such index) for which $P(t^*) = (\cdot, g)$ or $(g, \cdot)$; that is, P ends on either the right or the top boundary. Note that t^* can have any value in $[2g - 2]$. The number of possible monotone

staircase paths in a box $D_{i,j}$ is trivially bounded by $O(g^2 3^{2g-2})$, and by observing more carefully, we can bound it by $O(3^{2g})$, as is easily checked.[2]

We define the *cost* of a staircase path P in a box $D_{i,j}$ by

$$\mathrm{cost}_{i,j}(P) = \sum_{t=1}^{t^*} D_{i,j}(P(t)).$$

(For technical reasons, that will become clear in the sequel, we generally do not include the first position $P(0)$ of the path in evaluating its cost, except in the boxes $D_{i,1}$ and $D_{1,j}$ for all $i, j \in [s]$.) In the algorithm that follows, we want to assume (or ensure) that no two distinct paths in a box $D_{i,j}$ have the same cost. This will be the case if we assume that the input sequences are in sufficiently general position. We omit in this study perturbation techniques that can handle degenerate situations.

We denote by L the set of *positions* in the left and bottom boundaries of *any* box $D_{i,j}$, and by R the set of positions in the right and top boundaries (note that L and R have two common positions). Given a starting position $v \in L$, and an ending position $w \in R$, we denote by $S(v, w)$ the set of all staircase paths $P_{v,w}$ that start at v and end at w (if there is no staircase path between v and w, then $S(v, w) = \emptyset$). We say that $P^*_{v,w} \in S(v, w)$ is the *shortest path* between v and w in $D_{i,j}$ iff

$$\mathrm{cost}_{i,j}\left(P^*_{v,w}\right) = \min_{P_{v,w} \in S(v,w)} \left\{\mathrm{cost}_{i,j}\left(P_{v,w}\right)\right\}.$$

Note that according to our general position assumption, the shortest path between v and w, within a given box, is *unique*.

First Stage: Preprocessing. The first stage of our algorithm is to construct a data structure in subquadratic time (and storage), such that for each box $D_{i,j}$, and for each pair of positions $(v, w) \in L \times R$, we can retrieve the shortest path $P^*_{v,w}$ and $\mathrm{cost}_{i,j}(P^*_{v,w})$ in $O(1)$ time, when such a path exists (i.e., when $S(v, w)$ is nonempty).

The algorithm enumerates all $(2g-1)^2$ pairs of positions (v, w) in a $g \times g$ matrix (box) such that $v \in L$ and $w \in R$, discarding pairs that cannot be connected by a monotone staircase path, and referring to the surviving pairs as *admissible*. Again, we simplify the notation by upper bounding this quantity by $4g^2$. For each such admissible pair $(v, w) \in L \times R$, we enumerate *every* possible staircase path in $S(v, w)$ as $P_{v,w} : [t^*] \to [g] \times [g]$, where we write $P_{v,w} = \left(P^{\mathrm{r}}_{v,w}, P^{\mathrm{c}}_{v,w}\right)$ as a pair of row and column functions $P^{\mathrm{r}}_{v,w}$, $P^{\mathrm{c}}_{v,w} : [t^*] \to [g]$, so that $P_{v,w}(k) = \left(P^{\mathrm{r}}_{v,w}(k), P^{\mathrm{c}}_{v,w}(k)\right)$, for each $k \in [t^*]$. (Note that t^* is a path-dependent parameter, determined by v, w and the number of diagonal moves in the path.) There are $O(3^{2g})$ possible staircase paths $P_{v,w}$ (for all admissible pairs $(v, w) \in L \times R$ combined), so in total, we enumerate $O(3^{2g})$ staircase paths. These enumerations can be done in a natural lexicographic order, so that they induce an order on the $< 4g^2$ admissible pairs of positions of $L \times R$, and for each such pair (v, w), an order on all possible staircase paths $P_{v,w} \in S(v, w)$.

Given two staircase paths $P_{v,w}$ and $P'_{v,w}$ with the same starting and ending positions in a box $D_{i,j}$, we want to use the extended Fredman's trick (as in (3)) to compare $\mathrm{cost}_{i,j}\left(P_{v,w}\right)$ with $\mathrm{cost}_{i,j}\left(P'_{v,w}\right)$, by comparing two expressions such that one de-

[2] Each staircase path can be encoded by its first position, followed by its sequence of moves, where each move is in one of the directions up/right/up-right. Thus, the number of staircase paths that start in some position $(r, 1)$ (resp. $(1, r)$) on the left (resp. bottom) boundary is bounded by 3^{2g-1-r}. Thus, the total number of staircase paths that start in the left or the bottom boundary is bounded by $2\sum_{r=1}^{g} 3^{2g-1-r} = O(3^{2g})$.

pends on points from A_i only and the other depends on points from B_j only. Suppose that $P_{v,w} = ((\ell_1, m_1), \ldots, (\ell_r, m_r))$ and $P'_{v,w} = ((\ell'_1, m'_1), \ldots, (\ell'_t, m'_t))$ (note that $(\ell_r, m_r) = (\ell'_t, m'_t) = w$, since both paths end at w, and that we ignore the starting positions $(\ell_0, m_0) = (\ell'_0, m'_0) = v$). We have

$$\text{cost}_{i,j}(P_{v,w}) = \big|A_i(\ell_1) - B_j(m_1)\big| + \cdots + \big|A_i(\ell_r) - B_j(m_r)\big|,$$

and

$$\text{cost}_{i,j}\left(P'_{v,w}\right) = \big|A_i(\ell'_1) - B_j(m'_1)\big| + \cdots + \big|A_i(\ell'_t) - B_j(m'_t)\big|,$$

and we want to test whether, say, $\text{cost}_{i,j}(P_{v,w}) < \text{cost}_{i,j}\left(P'_{v,w}\right)$ (recall that we assume that equalities do not arise), that is, testing whether

$$\big|A_i(\ell_1)-B_j(m_1)\big|+\cdots+\big|A_i(\ell_r)-B_j(m_r)\big| < \big|A_i(\ell'_1)-B_j(m'_1)\big|+\cdots+\big|A_i(\ell'_t)-B_j(m'_t)\big|. \quad (4)$$

The last term in each side of (4) is actually unnecessary, since they are equal. In order to transform this inequality into a form suitable for applying the extended Fredman's trick (3), we need to replace each absolute value $|x|$ by either $+x$ or $-x$, as appropriate. To see what we are after, assume first that the expressions $A_i(\ell_k) - B_j(m_k)$ and $A_i(\ell'_k) - B_j(m'_k)$ are all positive, so that (4) becomes

$$A_i(\ell_1) - B_j(m_1) + \cdots + A_i(\ell_r) - B_j(m_r) < A_i(\ell'_1) - B_j(m'_1) + \cdots + A_i(\ell'_t) - B_j(m'_t).$$

By (3) we can rewrite this inequality as

$$A_i(\ell_1)+\cdots+A_i(\ell_r)-A_i(\ell'_1)-\cdots-A_i(\ell'_t) < B_j(m_1)+\cdots+B_j(m_r)-B_j(m'_1)-\cdots-B_j(m'_t),$$

which can be written as

$$A_i(P^{\mathrm{r}}_{v,w}(1)) + \cdots + A_i(P^{\mathrm{r}}_{v,w}(r)) - A_i(P'^{\,\mathrm{r}}_{v,w}(1)) - \cdots - A_i(P'^{\,\mathrm{r}}_{v,w}(t)) \quad (5)$$

$$< B_j(P^{\mathrm{c}}_{v,w}(1)) + \cdots + B_j(P^{\mathrm{c}}_{v,w}(r)) - B_j(P'^{\,\mathrm{c}}_{v,w}(1)) - \cdots - B_j(P'^{\,\mathrm{c}}_{v,w}(t)). \quad (6)$$

If $P_{v,w} = P^*_{v,w}$ (i.e., if $P_{v,w}$ is the shortest path from v to w) in $D_{i,j}$ then the inequality above holds for all pairs $(P_{v,w}, P'_{v,w})$, where $P'_{v,w} \in S(v,w)$ is any other staircase path between v and w.

For each admissible pair of positions $(v,w) \in L \times R$, we guess a staircase path $P_{v,w}$ as a candidate for being the shortest path from v to w. The overall number of such guesses is fewer than $(3^{2g})^{4g^2} = 3^{8g^3}$. For a fixed choice of paths, one for each admissible pair $(v,w) \in L \times R$, we want to test whether all the $< 4g^2$ guessed paths are the shortest paths between the corresponding pairs of positions. As unfolded next, we will apply this test for all boxes $D_{i,j}$, and output those boxes at which the outcome is positive (for the current guessed set of shortest paths). We will repeat the procedure for all $< 3^{8g^3}$ possible sets of guessed paths $P_{v,w}$.

Testing a fixed guess of shortest paths. For each group A_i, we create a (blue) point α_i, and for each group B_j we create a (red) point β_j, such that, for every admissible pair $(v,w) \in L \times R$, we have one coordinate for each path $P'_{v,w} \in S(v,w)$, different from the guessed path. The value of α_i (resp., β_j) at that coordinate is the corresponding expression (5) (resp., (6)). The points α_i and β_j are embedded in $\mathbb{R}^{d_g}$, where $d_g = \sum_{(v,w)} \Gamma_{v,w}$ is the sum over all admissible pairs $(v,w) \in L \times R$, and $\Gamma_{v,w}$ is the number of monotone staircase paths from v to w minus 1. As discussed earlier, $d_g = O(3^{2g})$.

We have that a (blue) point

$$\alpha_i = \left(\dots, A_i(P^{\mathrm{r}}_{v,w}(1)) + \dots + A_i(P^{\mathrm{r}}_{v,w}(r)) - A_i(P'^{\mathrm{r}}_{v,w}(1)) - \dots - A_i(P'^{\mathrm{r}}_{v,w}(t)), \dots\right)$$

is dominated by a (red) point

$$\beta_j = \left(\dots, B_j(P^{\mathrm{c}}_{v,w}(1)) + \dots + B_j(P^{\mathrm{c}}_{v,w}(r)) - B_j(P'^{\mathrm{c}}_{v,w}(1)) - \dots - B_j(P'^{\mathrm{c}}_{v,w}(t)), \dots\right),$$

if and only if each of the paths that we guessed (a path for every admissible pair $(v, w) \in L \times R$) are the shortest paths between the corresponding positions v, w in box $D_{i,j}$. The number of points is $2s = \Theta(n/g)$, and the time to prepare the points, i.e., to compute all their coordinates, is $O(2s \cdot 3^{2g} \cdot g) = O(3^{2g}n)$.

By Lemma 3, we can report all pairs of points (α_i, β_j) such that α_i is dominated by β_j, in $O\left(c_\varepsilon^{3^{2g}}(n/g)^{1+\varepsilon} + K\right)$ time, where K is the number of boxes at which the test of our specific guesses comes out positive. As mentioned earlier, we use $\varepsilon = 1/2$, with $c_\varepsilon \approx 3.42$.

This runtime is for a specific guess of a set of shortest paths between all admissible pairs in $L \times R$. As already mentioned, we repeat this procedure at most 3^{8g^3} times. Overall, we will report exactly $s^2 = \Theta\left((n/g)^2\right)$ dominating pairs (red on blue), because the set of shortest paths between admissible pairs in $L \times R$ in each box $D_{i,j}$ is unique (recall, we assumed that any pair of distinct staircase paths in a box do not have the same cost). Since the overall number of guesses is bounded by 3^{8g^3}, the overall runtime for all invocations of the bichromatic dominance reporting algorithm (including preparing the points) is

$$O\left(3^{8g^3}\left(3^{2g}n + c_\varepsilon^{3^{2g}}(n/g)^{1+\varepsilon}\right) + (n/g)^2\right).$$

Recall that, so far, we have assumed that all the differences within the absolute values $D_{i,j}[\ell, m] = \left|A_i(\ell) - B_j(m)\right|$ are positive, which allowed us to drop the absolute values, and write $D_{i,j}[\ell, m] = A_i(\ell) - B_j(m)$, for every $i, j \in [s]$, and $\ell, m \in [g]$, thereby facilitating the use of (the extended) Fredman's trick (3). Of course, in general this will not be the case, so, in order to still be able to drop the absolute values, we also have to guess the signs of all these differences.

For each box $D_{i,j}$, there is a *unique* sign assignment $\sigma^* : [g] \times [g] \to \{-1, 1\}$ such that

$$D_{i,j}[\ell, m] = \left|A_i(\ell) - B_j(m)\right| = \sigma^*(\ell, m)(A_i(\ell) - B_j(m)),$$

for every $\ell, m \in [g]$ (our "general position" assumption implies that each difference is nonzero). Thus for any staircase path $P = (P^{\mathrm{r}}, P^{\mathrm{c}})$ in $D_{i,j}$, of length t^*, we have

$$\mathrm{cost}_{i,j}(P) = \sum_{t=1}^{t^*} \sigma^*(P(t))\left(A_i(P^{\mathrm{r}}(t)) - B_j(P^{\mathrm{c}}(t))\right).$$

Now we proceed as before, guessing sets of paths, but now we also guess the sign assignment of the box, by trying *every* possible assignment $\sigma : [g] \times [g] \to \{-1, 1\}$, and modify the points α_i and β_j, defined earlier, by (i) adding sign factors to each term, and (ii) adding coordinates that enable us to test whether σ is the correct assignment σ^* for the corresponding boxes $D_{i,j}$.

Denote by P the guessed shortest path for some admissible pair of positions $(v, w) \in L \times R$, and let σ be the guessed sign assignment. Then, for every other path $P' \in S(v, w)$, we have the following modified coordinates for α_i and β_j respectively.

$$\left(\dots, \sigma(P(1))A_i(P^{\mathrm{r}}(1)) + \dots + \sigma(P(r))A_i(P^{\mathrm{r}}(r)) - \sigma(P'(1))A_i(P'^{\mathrm{r}}(1)) - \dots - \sigma(P'(t))A_i(P'^{\mathrm{r}}(t)), \dots\right),$$
$$\left(\dots, \sigma(P(1))B_j(P^{\mathrm{c}}(1)) + \dots + \sigma(P(r))B_j(P^{\mathrm{c}}(r)) - \sigma(P'(1))B_j(P'^{\mathrm{c}}(1)) - \dots - \sigma(P'(t))B_j(P'^{\mathrm{c}}(t)), \dots\right),$$

where we use the same notations as in (4), (5), and (6). In addition, to validate the correctness of σ, we extend α_i and β_j by adding the following g^2 coordinates to each of them. For every pair $(\ell, m) \in [g] \times [g]$, we add the following coordinates to α_i and β_j respectively.

$$(\ldots, -\sigma(\ell, m)A_i(\ell), \ldots),$$
$$(\ldots, -\sigma(\ell, m)B_j(m), \ldots).$$

This ensures that a point α_i is dominated by a point β_j if and only if $D_{i,j}[\ell, m] = \sigma(\ell, m)\left(A_i(\ell) - B_j(m)\right)$, for every $\ell, m \in [g]$, and all the $< 4g^2$ paths that we guessed are indeed shortest paths in box $D_{i,j}$.

The runtime analysis is similar to the preceding one, but now we increase the number of guesses by a factor of 2^{g^2} for the sign assignments, and the dimension of the space where the points are embedded increases by g^2 additional coordinates. We now have $2s = \Theta(n/g)$ points in $\mathbb{R}^{d_g+g^2}$ ($d_g = O(3^{2g})$ is as defined earlier), and the time to prepare them (computing the value of each coordinate) is $O((n/g)(d_g + g^2)g) = O(3^{2g}n)$. There are at most 3^{8g^3} sets of paths to guess, and for each set, there are at most 2^{g^2} sign assignment guesses, so in total, we invoke the bichromatic dominance reporting algorithm at most $2^{g^2}3^{8g^3} < 3^{8g^3+g^2}$ times, for an overall runtime (including preparing the points) of

$$O\left(3^{8g^3+g^2}\left(3^{2g}n + c_\varepsilon^{3^{2g}+g^2}(n/g)^{1+\varepsilon}\right) + (n/g)^2\right).$$

By setting $\varepsilon = 1/2$ and $g = \delta \log\log n$, for a suitable sufficiently small constant δ, the first two terms become negligible (strongly subquadratic), and the runtime is therefore dominated by the output size, that is $O\left((n/g)^2\right) = O\left(n^2/(\log\log n)^2\right)$. Each reported pair (α_i, β_j) certifies that the current set of $< 4g^2$ guessed paths are all shortest paths in box $D_{i,j}$. Each of the $s^2 = \Theta\left((n/g)^2\right)$ sets of shortest paths is represented by $O(g^3) = O((\log\log n)^3)$ bits (there are $< 4g^2$ shortest paths connecting admissible pairs, each of length at most $2g-1$, and each path can be encoded by its first position, followed by the sequence of its at most $2g-2$ moves, where each move is in one of the three directions up/right/up-right), and thus it can easily be stored in one machine word (for sufficiently small δ). Moreover, we have an order on the pairs (v, w) (induced by our earlier enumeration), so for each set, we can store its shortest paths in this order, and therefore, accessing a specific path (for some admissible pair) from the set takes $O(1)$ time.

Note, however, that we obtain only the *positions* that the paths traverse and not their *cost*. In later stages of our algorithm we will also need to compute, on demand, the cost of certain paths, but doing this naively would take $O(g)$ time per path, which is too expensive for us. To handle this issue, when we guess a sign assignment σ, and a set S of the $< 4g^2$ paths as candidates for the shortest paths, we also compute and store, for each path $P \in S$ that we have not yet encountered, the *rows-value* of P in A_i,

$$V_i^{\mathrm{r}}(P, \sigma) = \sigma(P(1))A_i(P^{\mathrm{r}}(1)) + \cdots + \sigma(P(t^*))A_i(P^{\mathrm{r}}(t^*)),$$

for every $i \in [s]$, and the *columns-value* of P in B_j,

$$V_j^{\mathrm{c}}(P, \sigma) = \sigma(P(1))B_j(P^{\mathrm{c}}(1)) + \cdots + \sigma(P(t^*))B_j(P^{\mathrm{c}}(t^*)),$$

for every $j \in [s]$, where t^* is the length of P. Observe that, for the correct sign assignment σ^* of box $D_{i,j}$,

$$\mathrm{cost}_{i,j}(P) = V_i^{\mathrm{r}}(P, \sigma^*) - V_j^{\mathrm{c}}(P, \sigma^*). \tag{7}$$

We do not compute $V_i^{\mathrm{r}}(P,\sigma) - V_j^{\mathrm{c}}(P,\sigma)$ yet, but only compute and store (if not already stored) the separate quantities $V_i^{\mathrm{r}}(P,\sigma)$ and $V_j^{\mathrm{c}}(P,\sigma)$, for each $P \in S$, for every guessed set S, and sign assignment σ. We store the values $V_i^{\mathrm{r}}(P,\sigma)$ and $V_j^{\mathrm{c}}(P,\sigma)$ in arrays, ordered by the earlier enumeration of all staircase paths, so that given a staircase path P, and indices $\kappa, \kappa' \in \left[\frac{n}{g-1}\right]$, we can retrieve, upon demand, the values $V_\kappa^{\mathrm{r}}(P,\sigma^*)$ and $V_{\kappa'}^{\mathrm{c}}(P,\sigma^*)$, and compute $\mathrm{cost}_{\kappa,\kappa'}(P)$ by using (7), in $O(1)$ time. In total, over all our guessed paths and sign assignments, this takes $O(2^{g^2}3^{2g} \cdot (n/g) \cdot g) = O(3^{g^2+2g}n)$ time and space, which is already subsumed by the time (and space) bound for reporting dominances from the previous stage.

To summarize this stage of the algorithm, we presented a subquadratic-time preprocessing procedure, which runs in $O\left((n/g)^2\right) = O\left(n^2/(\log\log n)^2\right)$ time, such that for any box $D_{i,j}$, and an admissible pair of positions $(v,w) \in L \times R$, we can retrieve the shortest path $P^*_{v,w}$ in $O(1)$ time, as well as compute $\mathrm{cost}_{i,j}(P^*_{v,w})$ in $O(1)$ time. This will be useful in the next stage of our algorithm.

Second Stage: Compact Dynamic Programming. Our approach is to view the $(n+1) \times (n+1)$ matrix M from the dynamic programming algorithm as decomposed into $s^2 = \left(\frac{n}{g-1}\right)^2$ boxes $M_{i,j}$, each of size $g \times g$, so that each box $M_{i,j}$ occupies the same positions as does the corresponding box $D_{i,j}$. That is, the indices of the rows (resp., columns) of $M_{i,j}$ are those of A_i (resp., B_j). In particular, for each $i,j \in [s]$, the positions $(\cdot, g)$ on the right boundary of each box $M_{i,j}$ coincide with the corresponding positions $(\cdot, 1)$ on the left boundary of $M_{i,j+1}$, and the positions $(g,\cdot)$ on the top boundary of $M_{i,j}$ coincide with the corresponding positions $(1,\cdot)$ on the bottom boundary of $M_{i+1,j}$. Formally, $M_{i,j}[\ell,m] = M\left[(i-1)(g-1)+\ell, (j-1)(g-1)+m\right]$, for each position $(\ell,m) \in [g] \times [g]$. See Figure 1 for an illustration.

Our strategy is to traverse the boxes, starting from the leftmost-bottom one $M_{1,1}$, where we already have the values of M at the positions of its left and bottom boundaries L, and we compute the values of M on its top and right boundaries R. We then continue to the box on the right, $M_{1,2}$, now having the values on its L-boundary (where its left portion overlaps with the R-boundary of $M_{1,1}$ and its bottom portion is taken from the already preset bottom boundary), and we compute the values of M on its R-boundary. We continue in this way until we reach the rightmost-bottom box $M_{1,s}$. We then continue in the same manner in the next row of boxes, starting at $M_{2,1}$ and ending at $M_{2,s}$, and keep going through the rows of boxes in order. The process ends once we compute the values of M on the R-boundary of the rightmost-top box $M_{s,s}$, from which we obtain the desired entry $M[n,n]$.

For convenience, we enumerate the positions in L as $L(1), \ldots, L(2g-1)$ in "clockwise" order, so that $L(1)$ is the rightmost-bottom position $(1,g)$, and $L(2g-1)$ is the leftmost-top position $(g,1)$. Similarly, we enumerate the positions of R by $R(1), \ldots, R(2g-1)$ in "counterclockwise" order, with the same starting and ending locations. Let $M_{i,j}(L) = \{M_{i,j}[L(1)], \ldots M_{i,j}[L(2g-1)]\}$ and $M_{i,j}(R) = \{M_{i,j}[R(1)], \ldots M_{i,j}[R(2g-1)]\}$, for $i,j \in [s]$.

By definition, for each position $(\ell,m) \in [n+1] \times [n+1]$, $M[\ell,m]$ is the minimal cost of a staircase path from $(0,0)$ to (ℓ,m). It easily follows, by construction, that for each box $D_{i,j}$, and for each $w \in R$, we have

$$M_{i,j}[w] = \min_{\substack{u \in L \\ (u,w) \text{ admissible}}} \Big\{ M_{i,j}[u] + \mathrm{cost}_{i,j}(P^*_{u,w}) \Big\}. \tag{8}$$

(Note that, by definition, the term $D_{i,j}[u]$ is included in $M_{i,j}[u]$ and not in $P^*_{u,w}$, so it is not doubly counted.) For each box $M_{i,j}$ and each position $w \in R$, our goal is thus to compute

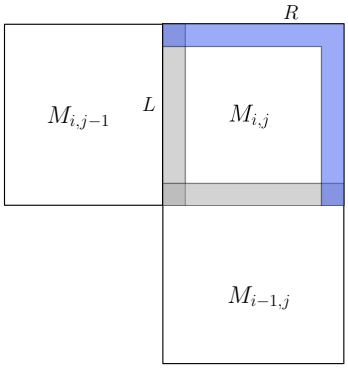


Figure 1 The L-boundary (shaded in gray) of box $M_{i,j}$ overlaps with the top boundary of $M_{i-1,j}$ and the right boundary of $M_{i,j-1}$. Once we have the values of M at the positions of the L-boundary of $M_{i,j}$, our algorithm computes the values of M at the positions of its R-boundary (shaded in blue).

the position $u \in L$ that attains the minimum in (8). We call such (u, w) the *minimal pair* for w in $M_{i,j}$.

For each box $D_{i,j}$, and each admissible pair $(v, w) \in L \times R$, we refer to the value $M_{i,j}[v] + \text{cost}_{i,j}(P^*_{v,w})$ as the *cumulative cost* of the pair (v, w), and denote it by c-cost(v, w).

We can rewrite (8), for each $w \in R$, as

$$M_{i,j}[w] = \min\{M^L_{i,j}[w],\ M^B_{i,j}[w]\},$$

where $M^B_{i,j}[w]$ is the minimum in (8) computed only over $u \in \{L(1), \ldots, L(g)\}$, which is the portion of L that overlaps the R-boundary of the bottom neighbor $M_{i-1,j}$ (when $i > 1$), and $M^L_{i,j}[w]$ is computed over $u \in \{L(g), \ldots, L(2g-1)\}$, which overlaps the R-boundary of the left neighbor $M_{i,j-1}$ (when $j > 1$). See Figure 1 for a schematic illustration. (Recall that the bottommost row and the leftmost column of M are initialized with ∞ values, except their shared cell $M[0, 0]$ that is initialized with 0.) The output of the algorithm is $M_{s,s}[R(g)] = M_{s,s}[g, g] = M[n, n]$. We can also return the optimal coupling, by using a simple backward pointer tracing procedure.

Computing minimal pairs. We still have to explain how to compute the minimal pairs (u, w) in each box $M_{i,j}$. Our preprocessing stage produces, for every box $D_{i,j}$, the set of all its shortest paths $S_{i,j} = \{P^*_{v,w} \mid (v, w) \in L \times R\}$ (ordered by the earlier enumeration of $L \times R$ and including only admissible pairs), and we can also retrieve the cost of each of these paths in $O(1)$ time (as explained earlier in the preprocessing stage). The cumulative cost (defined above) of each such pair (v, w) can also be computed in $O(1)$ time, assuming we have already computed $M_{i,j}[v]$. A naive, brute-force technique for computing the minimal pairs is to compute all the cumulative costs c-cost$_{i,j}(v, w)$, for all admissible pairs $(v, w) \in L \times R$, and select from them the minimal pairs. This however would take $O(g^2)$ time for each of the s^2 boxes, for a total of $\Theta(g^2 s^2) = \Theta(n^2)$ time, which is what we want to avoid.

Luckily, we have the following important lemma, which lets us compute all the minimal pairs within a box, significantly faster than in $O(g^2)$ time.

▶ **Lemma 4.** *For a fixed box $D_{i,j}$, and for any two distinct positions $w, w' \in R$, let $u, u' \in L$ be the positions for which (u, w) and (u', w') are minimal pairs in $M_{i,j}$. Then their corresponding shortest paths $P^*_{u,w}$ and $P^*_{u',w'}$ can partially overlap but can never cross each other. Formally, assuming that $w > w'$ (in the counterclockwise order along R), we have that for any*

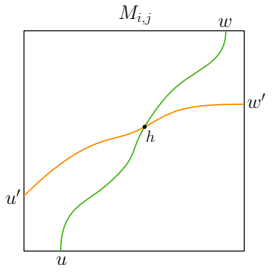


Figure 2 By Lemma 4, if (u, w) and (u', w') are minimal pairs in $M_{i,j}$, then the illustrated scenario is impossible, since the path $P^*_{u,w}$ (in green) is a portion of the shortest path from $M[0, 0]$ to $M_{i,j}[w]$, and the path $P^*_{u',w'}$ (in orange) is a portion of the shortest path from $M[0, 0]$ to $M_{i,j}[w']$. The illustrated intersection implies that one of the latter paths can decrease its cumulative cost by replacing its portion that ends at h by the respective portion that ends in h of the other path, which contradicts the fact that both of these paths are shortest paths.

$\ell, \ell', m \in [g]$, if $(\ell, m) \in P^*_{u,w}$ and $(\ell', m) \in P^*_{u',w'}$ then $\ell \geq \ell'$. That is, $P^*_{u,w}$ lies fully above $P^*_{u',w'}$ (partial overlapping is possible). In particular, we also have $u \geq u'$ (in the clockwise order along L)

Lemma 4 asserts the so-called *Monge property* of shortest-path matrices (see, e.g., [10, 24]). See Figure 2 for an illustration (of an impossible crossing) and a sketch of a proof.

We can therefore use the following divide-and-conquer paradigm for computing the minimal pairs within a box $D_{i,j}$. We start by setting the median index $k = \lfloor |R|/2 \rfloor$ of $|R|$, and compute the minimal pair $(u, R(k))$ and c-cost$(u, R(k))$, naively, in $O(g)$ time, as explained above. The path $P^*_{u,R(k)}$ decomposes the box $D_{i,j}$ into two parts, so that one part, X, consists all the positions in $D_{i,j}$ that are (weakly) *above* $P^*_{u,R(k)}$, and the other part, Y, consists all the positions in $D_{i,j}$ that are (weakly) *below* $P^*_{u,R(k)}$, so that X and Y are disjoint, except for the positions along the path $P^*_{u,R(k)}$ which they share. By Lemma 4, the shortest paths between any other minimal pair of positions in $L \times R$ can never cross $P^*_{u,R(k)}$. Thus, we can repeat this process separately in X and in Y. Note that the input to each recursive step is just the sequences of positions of X and Y along L and R, respectively (and we encode each sequence simply by its first and last elements); there is no need to keep track of the corresponding portion of $D_{i,j}$ itself.

Denote by $T(a, b)$ the maximum runtime for computing all the minimal pairs (u, w), within any box $M_{i,j}$, for u in some contiguous portion L' of a entries of L, and w in some contiguous portion R' of b entries of R. Clearly, $T(1, b) = O(b)$, and $T(a, 1) = O(a)$. In general, the runtime is bounded by the recurrence

$$T(a, b) = \max_{k \in [a]} \Big\{ T(k,\, b/2) + T(a - k + 1,\, b/2) \Big\} + O(a).$$

It is an easy exercise to show that the solution of this recurrence satisfies $T(a, b) = O\left((a + b) \log b\right)$. Thus, the runtime of the divide-and-conquer procedure described above, for a fixed box $M_{i,j}$, is $O\left((|R| + |L|) \log |R|\right) = O(g \log g)$.

The runtime of computing $M_{i,j}(R)$ for all $s^2 = \Theta\left((n/g)^2\right)$ boxes is thus $O\left((n/g)^2 g \log g\right) = O\left(n^2 \log g/g\right)$. Overall, including the preprocessing stage, the total runtime of the algorithm is $O\left((n/g)^2 + n^2 \log g/g\right) = O\left(n^2 \log g/g\right)$. As dictated by the preprocessing stage, we need to choose $g = \Theta(\log \log n)$, so the overall runtime is $O\left(n^2 \log \log \log n / \log \log n\right)$. This completes the proof of Theorem 1 for DTW. ◀

References

1. A. Abboud, A. Backurs, and V. V. Williams. Tight hardness results for lcs and other sequence similarity measures. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 59–78, 2015. doi:10.1109/FOCS.2015.14.
2. Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: Or: A polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 375–388, 2016. doi:10.1145/2897518.2897653.
3. Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014. doi:10.1137/130920526.
4. Pankaj K. Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. Approximating dynamic time warping and edit distance for a pair of point sequences. In *Proceedings of the 32nd International Symposium on Computational Geometry, SoCG*, pages 6:1–6:16, 2016. doi:10.4230/LIPIcs.SoCG.2016.6.
5. V. Arlazarov, E. Dinic, M. Kronrod, and I. Faradzev. On economical construction of the transitive closure of a directed graph. *Dokl. Akad. Nauk.*, 194(11), 1970.
6. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, STOC*, pages 51–58, 2015. doi:10.1145/2746539.2746612.
7. Philip Bille and Martin Farach-Colton. Fast and compact regular expression matching. *Theoretical Computer Science*, 409(3):486–496, 2008. doi:10.1016/j.tcs.2008.08.042.
8. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.
9. Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 79–97, 2015. doi:10.1109/FOCS.2015.15.
10. Rainer E. Burkard, Bettina Klinz, and Rüdiger Rudolf. Perspectives of Monge properties in optimization. *Discrete Applied Mathematics*, 70(2):95–161, 1996. doi:10.1016/0166-218X(95)00103-X.
11. E. G. Caiani, A. Porta, G. Baselli, M. Turiel, S. Muzzupappa, F. Pieruzzi, C. Crema, A. Malliani, and S. Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. In *Computers in Cardiology*, pages 73–76, 1998. doi:10.1109/CIC.1998.731723.
12. T. M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50(2):236–243, 2008.
13. Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and I know it's you!: Implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 987–996, 2012. doi:10.1145/2207676.2208544.
14. Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, New York, 1998.
15. Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision*, 27(3):203–216, 2007. doi:10.1007/s10851-006-0647-0.

16 M. L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci*, 1(4):355–361, 1976.

17 M. L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976.

18 Omer Gold and Micha Sharir. Improved bounds for 3SUM, k-SUM, and linear degeneracy. *CoRR*, abs/1512.05279, 2015.

19 Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *CoRR*, abs/1607.05994, 2016. URL: `http://arxiv.org/abs/1607.05994`.

20 Szymon Grabowski. New tabulation and sparse dynamic programming based techniques for sequence similarity problems. *Discrete Applied Mathematics*, 212:96–103, 2016. Stringology Algorithms. `doi:10.1016/j.dam.2015.10.040`.

21 A. Grønlund and S. Pettie. Threesomes, degenerates, and love triangles. In *Proceedings 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 621–630, 2014.

22 Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. `doi:10.1006/jcss.2000.1727`.

23 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. `doi:10.1006/jcss.2001.1774`.

24 Haim Kaplan, Shay Mozes, Yahav Nussbaum, and Micha Sharir. Submatrix maximum queries in Monge matrices and Monge partial matrices, and their applications. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 338–355, 2012.

25 Eamonn Keogh and Ann Chotirat Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005. `doi:10.1007/s10115-004-0154-9`.

26 Eamonn J. Keogh and Michael J. Pazzani. *Scaling up Dynamic Time Warping to Massive Datasets*, pages 1–11. Springer Berlin-Heidelberg, 1999. `doi:10.1007/978-3-540-48247-5_1`.

27 Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 285–289, 2000. `doi:10.1145/347090.347153`.

28 Theo Gasser Kongming Wang. Alignment of curves by dynamic time warping. *Annals of Statistics*, 25(3):1251–1276, 1997.

29 William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980. `doi:10.1016/0022-0000(80)90002-1`.

30 Meinard Müller. *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin-Heidelberg, 2007. `doi:10.1007/978-3-540-74048-3_4`.

31 F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer, New York, NY, 1985.

32 Chotirat A. Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 506–510, 2005. `doi:10.1137/1.9781611972757.50`.

33 T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57, 1968. `doi:10.1007/BF01074755`.

34 Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013. `doi:10.1007/s10618-012-0250-5`.

Efficient Construction of Probabilistic Tree Embeddings*†

Guy E. Blelloch[1], Yan Gu[2], and Yihan Sun[3]

1 **Carnegie Mellon University, Pittsburgh, PA, USA**
guyb@cs.cmu.edu

2 **Carnegie Mellon University, Pittsburgh, PA, USA**
yan.gu@cs.cmu.edu

3 **Carnegie Mellon University, Pittsburgh, PA, USA**
yihans@cs.cmu.edu

Abstract

In this paper we describe an algorithm that embeds a graph metric (V, d_G) on an undirected weighted graph $G = (V, E)$ into a distribution of tree metrics (T, D_T) such that for every pair $u, v \in V$, $d_G(u, v) \leq d_T(u, v)$ and $\mathbf{E}_T[d_T(u, v)] \leq O(\log n) \cdot d_G(u, v)$. Such embeddings have proved highly useful in designing fast approximation algorithms, as many hard problems on graphs are easy to solve on tree instances. For a graph with n vertices and m edges, our algorithm runs in $O(m \log n)$ time with high probability, which improves the previous upper bound of $O(m \log^3 n)$ shown by Mendel et al. in 2009.

The key component of our algorithm is a new approximate single-source shortest-path algorithm, which implements the priority queue with a new data structure, the *bucket-tree structure*. The algorithm has three properties: it only requires linear time in the number of edges in the input graph; the computed distances have a distance preserving property; and when computing the shortest-paths to the k-nearest vertices from the source, it only requires to visit these vertices and their edge lists. These properties are essential to guarantee the correctness and the stated time bound.

Using this shortest-path algorithm, we show how to generate an intermediate structure, the approximate dominance sequences of the input graph, in $O(m \log n)$ time, and further propose a simple yet efficient algorithm to converted this sequence to a tree embedding in $O(n \log n)$ time, both with high probability. Combining the three subroutines gives the stated time bound of the algorithm.

We also show a new application of probabilistic tree embeddings: they can be used to accelerate the construction of a series of approximate distance oracles.

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

Keywords and phrases Graph Algorithm, Metric Embeddings, Probabilistic Tree Embeddings, Single-source Shortest-paths

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.26

* The full version of this paper is available at arXiv:1605.04651 [7], https://arxiv.org/abs/1605.04651.

† This research was supported in part by NSF grants CCF-1314590 and CCF-1533858, and the Intel Science and Technology Center for Cloud Computing.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 26; pp. 26:1–26:14

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

1 Introduction

The idea of probabilistic tree embeddings [4] is to embed a finite metric into a distribution of tree metrics with a minimum expected distance distortion. A distribution $\mathcal{D}$ of trees of a metric space (X, d_X) should minimize the expected stretch ψ so that:

1. dominating property: for each tree $T \in \mathcal{D}$, $d_X(x, y) \leq d_T(x, y)$ for every $x, y \in X$, and
2. expected stretch bound: $\mathbf{E}_{T\sim\mathcal{D}}[d_T(x, y)] \leq \psi \cdot d_X(x, y)$ for every $x, y \in X$,

where $d_T(\cdot, \cdot)$ is the tree metric, and $\mathbf{E}_{T\sim\mathcal{D}}$ draws a tree T from the distribution $\mathcal{D}$. After a sequence of results [2, 3, 4], Fakcharoenphol, Rao and Talwar [17] eventually proposed an elegant and asymptotically optimal algorithm (FRT-embedding) with $\psi = O(\log n)$.

Probabilistic tree embeddings facilitate many applications. They lead to practical algorithms to solve a number of problems with good approximation bounds, for example, the k-median problem, buy-at-bulk network design [8], and network congestion minimization [30]. A number of network algorithms use tree embeddings as key components, and such applications include generalized Steiner forest problem, the minimum routing cost spanning tree problem, and the k-source shortest paths problem [22]. Also, tree embeddings are used in solving symmetric diagonally dominant (SDD) linear systems. Classic solutions use spanning trees as the preconditioner, but recent work by Cohen et al. [14] describes a new approach to use trees with Steiner nodes (e.g. FRT trees).

In this paper we discuss yet another remarkable application of probabilistic tree embeddings: constructing of approximate distance oracles (ADOs)—a data structure with compact storage ($o(n^2)$) which can approximately and efficiently answer pairwise distance queries on a metric space. We show that FRT trees can be used to accelerate the construction of some ADOs [24, 34, 10].

Motivated by these applications, efficient algorithms to construct tree embeddings are essential, and there are several results on the topic in recent years [12, 22, 8, 25, 21, 19, 6]. Some of these algorithms are based on different parallel settings, e.g. share-memory setting [8, 19, 6] or distributed setting [22, 21]. As with this paper, most of these algorithms [12, 22, 25, 21, 6] focus on graph metrics, which most of the applications discussed above are based on. In the sequential setting, i.e. on a RAM model, to the best of our knowledge, the most efficient algorithm to construct optimal FRT-embeddings was proposed by Mendel and Schwob [25]. It constructs FRT-embeddings in $O(m \log^3 n)$ expected time given an undirected positively weighted graph with n vertices and m edges. This algorithm, as well as the original construction in the FRT paper [17], works hierarchically by generating each level of a tree top-down. However, such a method can be expensive in time and/or coding complexity. The reason is that the diameter of the graph can be arbitrarily large and the FRT trees may contain many levels, which requires complicated techniques, such as building sub-trees based on quotient graphs.

Our results. The main contribution of this paper is an efficient construction of the FRT-embeddings. Given an undirected positively weighted graph $G = (V, E)$ with n vertices and m edges, our algorithm builds an optimal tree embedding in $O(m \log n)$ time. In our algorithm, instead of generating partitions by level, we adopt an alternative view of the FRT algorithm in [22, 8], which computes the potential ancestors for each vertex using ***dominance sequences*** of a graph (first proposed in [12], and named as least-element lists in [12, 22]). The original algorithm to compute the dominance sequences requires $O(m \log n + n \log^2 n)$ time [12]. We then discuss a new yet simple algorithm to convert the dominance sequences to an FRT tree only using $O(n \log n)$ time. A similar approach was taken by Khan et al. [22] but their output is an implicit representation (instead of an tree) and under the distributed

setting and it is not work-efficient without using the observations and tree representations introduced by Blelloch et al. in [8].[1]

Based on the algorithm to efficiently convert the dominance sequences to FRT trees, the time complexity of FRT-embedding construction is bottlenecked by the construction of the dominance sequences. Our efficient approach contains two subroutines:

- An efficient (approximate) single-source shortest-path algorithm, introduced in Section 3. The algorithm has three properties: linear complexity, distance preservation, and the ordering property (full definitions given in Section 3). All three properties are required for the correctness and efficiency of constructing FRT-embedding. Our algorithm is a variant of Dijkstra's algorithm with the priority queue implemented by a new data structure called *bucket-tree structure*.
- An algorithm to integrate the shortest-path distances into the construction of FRT trees, discussed in Section 4. When the diameter of the graph is $n^{O(1)}$, we show that an FRT tree can be built directly using the approximate distances computed by shortest-path algorithm. The challenge is when the graph diameter is large, and we proposed an algorithm that computes the approximate dominance sequences of a graph by concatenating the distances that only use the edges within a relative range of $n^{O(1)}$. Then we show why the approximate dominance sequences still yield valid FRT trees.

With these new algorithmic subroutines, we show that the time complexity of computing FRT-embedding can be reduced to $O(m \log n)$ w.h.p. for an undirected positively weighted graph with arbitrary edge weight.

In addition to the efficient construction of FRT trees, this paper also discuss a new application. We show that FRT trees are intrinsically Ramsey partitions (definition given in Section 5) with asymptotically tight bound, and can achieve even better (constant) bounds on distance approximation. Previous construction algorithms of optimal Ramsey partitions are based on hierarchical CKR partitions, namely, on each level, the partition is individually generated with an independent random radius and new random priorities. In this paper, we present a new proof to show that the randomness in each level is actually unnecessary, so that only one single random permutation is enough and the ratio of radii in consecutive levels can be fixed as 2. Our FRT-tree construction algorithm therefore can be directly applied to a number of different distance oracles that are based on Ramsey partitions and accelerates the construction of these distance oracles.

2 Preliminaries and Notations

Let $G = (V, E)$ be a weighted graph with edge lengths $l : E \to \mathbb{R}_+$, and $d(u, v)$ denote the shortest-path distance in G between nodes u and v. Throughout this paper, we assume that $\min_{x \neq y} d(x, y) = 1$. Let $\Delta = \frac{\max_{x,y} d(x,y)}{\min_{x \neq y} d(x,y)} = \max_{x,y} d(x, y)$, the diameter of the graph G.

In this paper, we use the single source shortest paths problem (SSSP) as a subroutine for a number of algorithms. Consider a weighted graph with n vertices and m edges, Dijkstra's algorithm [15] solves the SSSP in $O(m + n \log n)$ time if the priority queue of distances is maintained using a Fibonacci heap [18].

A premetric (X, d_X) defines on a set X and provides a function $d : X \times X \to \mathbb{R}$ satisfying $d(x, x) = 0$ and $d(x, y) \geq 0$ for $x, y \in X$. A metric (X, d_X) further requires $d(x, y) = 0$ iff $x = y$, symmetry $d(x, y) = d(y, x)$, triangle inequality $d(x, y) \leq d(x, z) + d(z, y)$ for

[1] A simultaneous work by Friedrichs et al. proposed an $O(n \log^3 n)$ algorithm of this conversion (Lemma 7.2 in [19]).

$x, y, z \in X$. The shortest-path distances on a graph is a metric and is called the graph metric and denoted as d_G.

We assume all intermediate results of our algorithm have word size $O(\log n)$ and basic algorithmic operations can be finished within a constant time. Then within the range of $[1, n^k]$, the integer part of natural logarithm of an integer and floor function of an real number can be computed in constant time for any constant k. This can be achieved using standard table-lookup techniques (similar approaches can be found in Thorup's algorithm [32]). The time complexity of the algorithms are measured using the random-access machine (RAM) model.

A result holds *with high probability* (**w.h.p.**) for an input of size n if it holds with probability at least $1 - n^{-c}$ for any constant $c > 0$, over all possible random choices made by the algorithm.

Let $[n] = \{1, 2, \cdots, n\}$ where n is a positive integer.

We recall a useful fact about random permutations [31]:

▶ **Lemma 1.** *Let $\pi : [n] \to [n]$ be a permutation selected uniformly at random on $[n]$. The set $\{i \mid i \in [n], \pi(i) = \min\{\pi(j) \mid j = 1, \cdots, i\}\}$ contains $O(\log n)$ elements both in expectation and with high probability.*

3 An Approximate SSSP Algorithm

In this section we introduce a variant of Dijkstra's algorithm. This is an efficient algorithm for single-source shortest paths (SSSP) with linear time complexity $O(m)$. The computed distances are α-distance preserving:

▶ **Definition 2** (α-distance preserving)**.** For a weighted graph $G = (V, E)$, the single-source distances $d(v)$ for $v \in V$ from the source node s is α-distance preserving, if there exists a constant $0 \leq \alpha \leq 1$ such that $\alpha\, d_G(s, u) \leq d(u) \leq d_G(s, u)$, and $d(v) - d(u) \leq d_G(u, v)$, for every $u, v \in V$.

α-distance preserving can be viewed as the triangle inequality on single-source distances (i.e. $d(u) + d_G(u, v) \geq d(v)$ for $u, v \in V$), and is required in many applications related to distances. For example, in Corollary 4 we show that using Gabow's scaling algorithm [20] we can compute a $(1+\epsilon)$-approximate SSSP using $O(m \log \epsilon^{-1})$ time. Also in many metric problems including the contruction of optimal tree embeddings, distance preservation is necessary in the proof of the expected stretch, and such an example is Lemma 11 in Section 4.3.

The preliminary version we discussed in Section 3.1 limits edge weights in $[1, n^k]$ for a constant k, but with some further analysis in the full version of this paper we can extend the range to $[1, n^{O(m)}]$. This new algorithm also has two properties that are needed in the construction of FRT trees, while no previous algorithms achieve them all:

1. (α-distance preserving) The computed distances from the source $d(\cdot)$ is α-distance preserving.
2. (Ordering property and linear complexity) The vertices are visited in order of distance $d(\cdot)$, and the time to compute the first k distances is bounded by $O(m')$ where m' is the sum of degrees from these k vertices.

The algorithm also works on directed graphs, although this is not used in the FRT construction.

Approximate SSSP algorithms are well-studied [32, 23, 13, 29, 26]. In the sequential setting, Thorup's algorithm [32] compute single-source distances on undirected graphs with

integer weights using $O(n+m)$ time. Nevertheless, Thorup's algorithm does not obey the ordering property since it uses a hierarchical bucketing structure and does not visit vertices in an order of increasing distances, and yet we are unaware of a simple argument to fix this. Other algorithms are either not work-efficient (i.e. super-linear complexity) in sequential setting, and / or violating distance preservation.

▶ **Theorem 3.** *For a weighted directed graph $G=(V,E)$ with edge weights between 1 and $n^{O(1)}$, a $(1/4)$-distance preserving single-source shortest-path distances $d(\cdot)$ can be computed, such that the distance to the k-nearest vertices v_1 to v_k by $d(\cdot)$requires $O(\sum_{i=1}^{k}\text{degree}(v_i))$ time.*

The algorithm also has the two following properties. Since they are not used in the construction of FRT trees, we review them in the full version of this paper. We discuss how to (1) extend the range of edge weights to $n^{O(m)}$, and the cost to compute the k-nearest vertices is $O(\log_n d(v_k)+\sum_{i=1}^{k} degree(v_i))$ where v_1 to v_k are the k nearest vertices; and (2) compute $(1+\epsilon)$-distance-preserving shortest-paths for an arbitrary $\epsilon>0$:

▶ **Corollary 4.** *$(1+\epsilon)$-distance-preserving shortest-paths for all vertices can be computed by repeatedly using Theorem 3 $O(\log \epsilon^{-1})$ times.*

3.1 Algorithm Details

The key data structure in this algorithm is a bucket-tree structure shown in Figure 1 that implements the priority queue in Dijkstra's algorithm. With the bucket-tree structure, each Decrease-Key or Extract-Min operation takes constant time. Given the edge range in $[1,n^k]$, this structure has $l=\lceil(1+k)\log_2 n\rceil$ levels, each level containing a number of buckets corresponding to the distances to the source node. In the lowest level (level 1) the difference between two adjacent buckets is 2.

At anytime only one of the buckets in each level can be non-empty: there are in total l active buckets to hold vertices, one in each level. The active bucket in each level is the left-most bucket whose distance is larger than that of the current vertex being visited in our algorithm. We call these active buckets the **frontier** of the current distance, and they can be computed by the **path string**, which is a 0/1 bit string corresponding to the path from the current location to higher levels (until the root), and 0 or 1 is decided by whether the node is the left or the right child of its parent. For clarity, we call the buckets on the frontier **frontier buckets**, and the ancestors of the current bucket **ancestor buckets** (can be traced using the path string). For example, as the figure shows, if the current distance is 4, then the available buckets in the first several levels are the buckets corresponding to the distances $6,5,11,7$, and so on. The ancestor bucket and the frontier bucket in the same level may or may not be the same, depending on whether the current bucket is the left or right subtree of this bucket. For example, the path string for the current bucket with label 4 is 0100 and so on, and ancestor buckets correspond to $4,5,3,7$ and so on. It is easy to see that given the current distance, the path string, the ancestor buckets, and the frontier buckets can be computed in $O(l)$ time—constant time per level.

Note that since only one bucket in each level is non-empty, the whole structure need not to be build explicitly: we store one linked list for each level to represent the only active bucket in the memory (l lists in total), and use the current distance and path string to retrieve the location of the current bucket in the structure.

With the bucket-tree structure acting as the priority queue, we can run standard Dijkstra's algorithm. The only difference is that, to achieve linear cost for an SSSP query, the operations of Decrease-Key and Extract-Min need to be redefined on the bucket-tree structure.

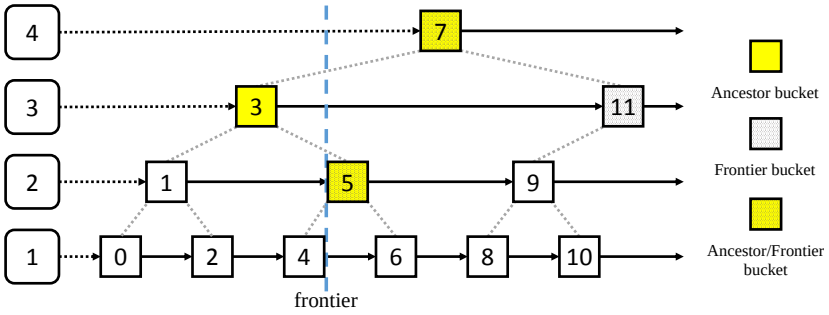


Figure 1 An illustration of a bucket-tree structure with the lowest 4 levels, and the current visiting bucket has distance 4. Notice that our algorithm does not insert vertices to the same level as the current bucket (i.e. bucket 6).

Once the relaxation of an edge succeeds, a DECREASE-KEY operation for the corresponding vertex will be applied. In the bucket-tree structure it is implemented by a DELETE (if the vertex is added before) followed by an INSERT on two frontier buckets respectively. The deletion is trivial with a constant cost, since we can maintain the pointer from each vertex to its current location in the bucket tree. We mainly discuss how to insert a new tentative distance into the bucket tree. When vertex u successfully relaxes vertex v with an edge e, we first round down the edge weight w_e by computing $r = \lfloor \log_2 (w_e + 1) \rfloor$. Then we find the appropriate frontier bucket B that the difference of the distances w'_e between this bucket B and the current bucket is the closest to (but no more than) $w_r = 2^r - 1$, and insert the relaxed vertex into this bucket. The constant approximation for this insertion operation holds due to the following lemma:

▶ **Lemma 5.** *For an edge with length w_e, the approximated length w'_e, which is the distance between the inserted bucket B and the current bucket, satisfies the following inequality: $w_e/4 \leq w'_e \leq w_e$.*

Proof. After the rounding, $w_r = 2^r - 1 = 2^{\lfloor \log_2 (w_e+1) \rfloor} - 1$ falls into the range of $[w_e/2, w_e]$. We now show that there always exists such a bucket B on the frontier that the approximated length w'_e is in $[w_r/2, w_r]$.

We use Algorithm 1 to select the appropriate bucket for a certain edge, given the current bucket level and the path string. The first case is when b, the current level, is larger than r. In this case all the frontier buckets on the bottom r levels form a left spine of the corresponding subtree rooted by the right child of the current bucket, so picking bucket in the r-th level leads to $w'_e = 2^{r-1}$, and therefore $w_e/4 < w'_e \leq w_e$ holds. The second case is when $b \leq r$, and the selected bucket is decided based on the structure on the ancestor buckets from the $(r+1)$-th level to $(r-1)$-th level, which is one of the three following cases.

- The simplest case ($b < r$, line 9) is when the ancestor bucket in the $(r-1)$-th level is the right child of the bucket in the r-th level. In this case when we pick the bucket in level r since the distance between two consecutive buckets in level r is 2^r, and the distance from the current bucket to the ancestor bucket in r-th level is at most $\sum_{i=1}^{r-1} 2^{i-1} < 2^{r-1}$. The distance thus between the current bucket and the frontier bucket in level r is $w'_e > 2^r - 2^{r-1} = 2^{r-1} > w_e/4$.
- The second case is when either $b = r$ and the current bucket is the left child (line 5), or $b < r$ and the ancestor bucket in level $r-1$ is on the left spine of the subtree rooted at the ancestor bucket in level $r+1$ (line 11). Similar to the first case, picking the frontier

Algorithm 1: Finding the appropriate bucket

Input: Current bucket level b, rounded edge length $2^r - 1$ and path string.
Output: The bucket in the frontier (the level is returned).

```
Let r' be the lowest ancestor bucket above level r that is a left child
if b > r then
    return r
else if b = r then
    if current bucket is left child then return r + 1
    else return r' + 1
else
    switch the branches from (r + 1)-th level to (r − 1)-th level in the path string do
        case left-then-right or right-then-right do
            return r
        case left-then-left do
            return r + 1
        case right-then-left do
            return r' + 1
```

bucket in the $(r+1)$-th level (which is also an ancestor bucket) skips the right subtree of the bucket in r-th level, which contains $2^{r-1} - 1 \geq w_e/4$ nodes.

- The last case is the same as the second case expect that the level-r ancestor bucket is the right child of level-$(r+1)$ ancestor bucket. In this case we will pick the frontier bucket that has distance 2^{r-1} to the ancestor bucket in level r, which is the parent of the lowest ancestor bucket that is a left child and above level r. In this case the approximated edge distance is between 2^{r-1} and $2^r - 1$.

Combining all these cases proves the lemma. ◀

We now explain ehe Extract-Min operation on the bucket tree. We will visit vertex in the current buckets one by one, so each Extract-Min has a constant cost. Once the traversal is finished, we need to find the next closest non-empty frontier.

▶ **Lemma 6.** Extract-Min *and* Decrease-Key *on the bucket tree require* $O(1)$ *time.*

Proof. We have shown that the modification on the linked list for each operation requires $O(1)$ time. A naïve implementation to find the bucket in Decrease-Key and Extract-Min takes $O(l) = O(\log n)$ time, by checking all possible frontier buckets. We can accelerate this look-up using the standard table-lookup technique. The available combinations of the input of Decrease-Key are n^{k+1} (total available current distance) by $l = O(k \log n)$ (total available edge distance after rounding), and the input combinations of Extract-Min are two $\lceil \log_2 n^{k+1} \rceil$ bit strings corresponding to the path to the root and the emptiness of the buckets on the frontier. We therefor partition the bucket tree into several parts, each containing $\lfloor (1-\epsilon')(\log_2 n)/2 \rfloor$ consecutive levels (for any $0 < \epsilon' < 1$). We now precompute the answer for all possible combinations of path strings and edge lengths, and (1) the sizes of look-up tables for both operations to be $O((2^{\lfloor (1-\epsilon')(\log_2 n)/2 \rfloor})^2) = o(n)$, (2) the cost for brute-force preprocessing to be $O((2^{\lfloor (1-\epsilon')(\log_2 n)/2 \rfloor})^2 \log n) = o(n)$, and (3) the time of either operation of Decrease-Key and Extract-Min to be $O(k)$, since each operation requires to look up at most $l/\lfloor (1-\epsilon')(\log_2 n)/2 \rfloor = O(k)$ tables. Since k is a constant, each of the two operations as well takes constant time. The update of path string can be computed similarly using this

table-lookup approach. As a result, with $o(n)$ preprocessing time, finding the associated bucket for DECREASE-KEY or EXTRACT-MIN operation uses $O(1)$ time. ◀

We now show the three properties of the new algorithm: linear complexity, triangle inequality, and the ordering property.

Proof of Theorem 3. Here we show the algorithm satisfies the properties in Theorem 3. Lemma 6 proves the linear cost of the algorithm. Lemma 5 shows that the final distances is α-distance preserving. Lastly, since this algorithm is actually a variant of Dijkstra's algorithm with the priority implemented by the bucket-tree structure, the ordering property is met, although here the k-nearest vertices are based on the approximate distances instead of real distances. ◀

In the full version we discuss how to extend the range of edge weight to $[1, n^{O(m)}]$.

4 The Dominance Sequence

In this section we review and introduce the notion of dominance sequences for each point of a metric space and describe the algorithm for constructing them on a graph. The basic idea of dominance sequences was previously introduced in [12] and [8]. Here we name the structure as the dominance sequence since the "dominance" property introduced below is crucial and related to FRT construction. In the next section we show how they can easily be converted into an FRT tree.

4.1 Definition

▶ **Definition 7** (Dominance)**.** Given a premetric (X, d_X) and a permutation π, for two points $x, y \in X$, x *dominates* y if and only if

$$\pi(x) = \min\{\pi(w) \mid w \in X, d_X(w, y) \leq d_X(x, y)\}.$$

Namely, x dominates y iff x's priority is greater (position in the permutation is earlier) than any point that is closer to y.

The dominance sequence for a point $x \in X$, is the sequence of all points that dominate x sorted by distance. More formally:

▶ **Definition 8** (Dominance Sequence[2])**.** For each $x \in X$ in a premetric (X, d_X), the *dominance sequence* of a point x with respect to a permutation $\pi : X \to [n]$ (denoted as $\chi_\pi^{(x)}$), is the sequence $\langle p_i \rangle_{i=1}^k$ such that $1 = \pi(p_1) < \pi(p_2) < \cdots < \pi(p_k) = \pi(x)$, and p_i is in $\chi_\pi^{(x)}$ iff p_i dominates x.

We use χ_π to refer to all dominance sequences for a premetric under permutation π. It is not hard to bound the size of the dominance sequence:

▶ **Lemma 9** ([13])**.** *Given a premetric (X, d_X) and a random permutation π, for each vertex $x \in X$, with w.h.p.*

$$\left|\chi_\pi^{(x)}\right| = O(\log n)$$

[2] Also called as "least-element list" in [12]. We rename it since in later sections we also consider many other variants of it based on the dominance property.

Algorithm 2: Efficient FRT tree construction

1 Pick a uniformly random permutation $\pi : V \to [n]$.
2 Compute the dominance sequences χ_π.
3 Pick $\beta \in [1, 2]$ with the probability density function $f_B(x) = 1/(x \ln 2)$.
4 Convert the dominance sequence χ_π to the compressed partition sequence $\bar{\sigma}_{\pi,\beta}$.
5 Generate the FRT tree based on $\bar{\sigma}_{\pi,\beta}$.

and hence overall, with w.h.p.

$$|\chi_\pi| = \sum_{x \in X} \left|\chi_\pi^{(x)}\right| = O(n \log n)$$

Since the proof is fairly straight-forward, for completeness we also provide it in the full version of this paper.

Now consider a graph metric (V, d_G) defined by an undirected positively weighted graph $G = (V, E)$ with n vertices and m edges, and $d_G(u, v)$ is the shortest distance between u and v on G. The dominance sequences of this graph metric can be constructed using $O(m \log n + n \log^2 n)$ time w.h.p. [12]. This algorithm is based on Dijkstra's algorithm.

4.2 Efficient FRT tree construction based on the dominance sequences

We now consider the construction of FRT trees based on a pre-computed dominance sequences of a given metric space (X, d_X). We assume the weights are normalized so that $1 \le d_X(x, y) \le \Delta = 2^\delta$ for all $x \neq y$, where δ is a positive integer.

The FRT algorithm [17] generates a top-down recursive low-diameter decomposition (LDD) of the metric, which preserves the distances up to $O(\log n)$ in expectation. It first chooses a random β between 1 and 2, and generates $1 + \log_2 \Delta$ levels of partitions of the graph with radii $\{\beta\Delta, \beta\Delta/2, \beta\Delta/4, \cdots\}$. This procedure produces a laminar family of clusters, which are connected based on set-inclusion to generate the FRT tree. The weight of each tree edge on level i is $\beta\Delta/2^i$.

Instead of computing these partitions directly, we adopt the idea of a point-centric view proposed in [8]. We use the intermediate data structure "dominance sequences" as introduced in Section 4.1 to store the useful information for each point. Then, an FRT tree can be retrieved from this sequence with very low cost:

▶ **Lemma 10.** *Given β and the dominance sequences χ_π of a metric space with associated distances to all elements, an FRT tree can be constructed using $O(n \log n)$ time w.h.p.*

The difficulty in this process is that, since the FRT tree has $O(\log \Delta)$ levels and Δ can be large (i.e. $\Delta > 2^{O(n)}$), an explicit representation of the FRT tree can be very costly. Instead we generate the compressed version with nodes of degree two removed and their incident edge weights summed into a new edge. The algorithm is outlined in Algorithm 2.

Proof. We use the definition of partition sequence and compressed partition sequence from [8]. Given a permutation π and a parameter β, the *partition sequence* of a point $x \in X$, denoted by $\sigma_{\pi,\beta}^{(x)}$, is the sequence $\sigma_{\pi,\beta}^{(x)}(i) = \min\{\pi(y) \mid y \in X, d(x, y) \le \beta \cdot 2^{\delta - i}\}$ for $i = 0, \ldots, \delta$, i.e. point y has the highest priority among vertices up to level i. We note that a trie (radix tree) built on the partition sequence is the FRT tree, but as mentioned we cannot build this explicitly. The compressed partition sequence, denoted as $\bar{\sigma}_{\pi,\beta}^{(x)}$, replaces consecutive

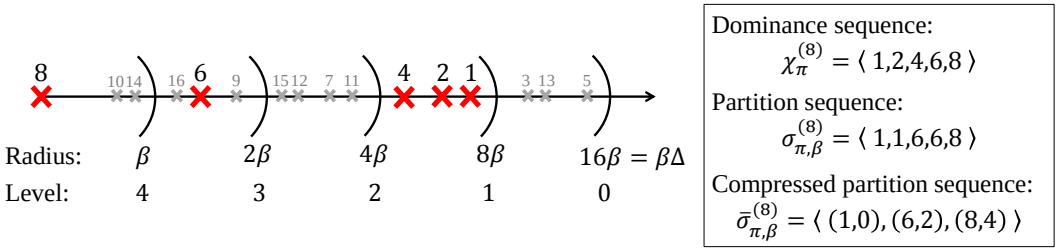


Figure 2 An illustration for dominance sequence, partition sequence and compressed partition sequence for vertex 8. Here we assume that the label of each vertex corresponds to its priority. The left part shows the distances of all vertices to vertex 8 in log-scale, and the red vertices dominate vertex 8.

equal points in the partition sequence $\sigma_{\pi,\beta}^{(x)}$ by the pair (p_i, l_i) where p_i is the vertex and l_i is the highest level p_i dominates x in the FRT tree. Figure 2 gives an example of a partition sequence, a compressed partition sequence, and their relationship to the dominance sequence.

To convert the dominance sequences χ_π to the compressed partition sequences $\bar{\sigma}_{\pi,\beta}$ note that for each point x the points in $\bar{\sigma}_{\pi,\beta}^{(x)}$ are a subsequence of $\chi_\pi^{(x)}$. Therefore, for $\bar{\sigma}_{\pi,\beta}^{(x)}$, we only keep the highest priority vertex in each level from $\chi_\pi^{(x)}$ and tag it with the appropriate level. Since there are only $O(\log n)$ vertices in $\chi_\pi^{(x)}$ w.h.p., the time to generate $\bar{\sigma}_{\pi,\beta}^{(x)}$ is $O(\log n)$ w.h.p., and hence the overall construction time is $O(n \log n)$ w.h.p.

The compressed FRT tree can be easily generated from the compressed partition sequences $\bar{\sigma}_{\pi,\beta}$. Blelloch et. al. [8] describe a parallel algorithm that runs in $O(n^2)$ time (sufficient for their purposes) and polylogarithmic depth. Here we describe a similar version to generate the FRT tree sequentially in $O(n \log n)$ time w.h.p. The idea is to maintain the FRT as a patricia trie [27] (compressed trie) and insert the compressed partition sequences one at a time. Each insertion just needs to follow the path down the tree until it diverges, and then either split an edge and create a new node, or create a new child for an existing node. Note that a hash table is required to trace the tree nodes since the trie has a non-constant alphabet. Each insertion takes time at most the sum of the depth of the tree and the length of the sequence, giving the stated bounds. ◀

We note that for the same permutation π and radius parameter β, it generates exactly the same tree as the original algorithm in [17].

4.3 Expected Stretch Bound

In Section 4.2 we discussed the algorithm to convert the dominance sequences to a FRT tree. When the dominance sequences is generated from a graph metric (G, d_G), the expected stretch is $O(\log n)$, which is optimal, and the proof is given in many previous papers [17, 8]. Here we show that any distance function $\hat{d}_G$ in Lemma 11 is sufficient to preserve this expected stretch. As a result, we can use the approximate shortest-paths computed in Section 3 to generate the dominance sequences and further convert to optimal tree embeddings.

▶ **Lemma 11.** *Given a graph metric (G, d_G) and a distance function $\hat{d}_G(u, v)$ such that for $u, v, w \in V$, $|\hat{d}_G(u, v) - \hat{d}_G(u, w)| \leq 1/\alpha \cdot d_G(v, w)$ and $d_G(u, v) \leq \hat{d}_G(u, v) \leq 1/\alpha \cdot d_G(u, v)$ for some constant $0 < \alpha \leq 1$, then the dominance sequences based on $(G, \hat{d}_G)$ can still yield optimal tree embeddings.*

Proof Outline. Since the overestimate distances hold the dominating property of the tree embeddings, we show the expected stretch is also not affected. We now show the expected stretch is also held.

Recall the proof of the expected stretch by Blelloch et al. in [8] (Lemma 3.4). By replacing d_G by $\hat{d}_G$, the rest of the proof remains unchanged except for Claim 3.5, which upper bounds the expected cost of a common ancestor w of $u, v \in V$ in u and v's dominance sequences. The original claim indicates that the probability that u and v diverges in a certain level centered at vertex w is $O(|d_G(w,u) - d_G(w,v)|/d_G(u,w)) = O(d_G(u,v)/d_G(u,w))$ and the penalty is $O(d_G(u,w))$, and therefore the contribution of the expected stretch caused by w is the product of the two, which is $O(d_G(u,v))$ (since there are at most $O(\log n)$ of such w (Lemma 9), the expected stretch is thus $O(\log n)$). With the distance function $\hat{d}_G$ and α as a constant, the probability now becomes $O(|\hat{d}_G(w,u) - \hat{d}_G(w,v)|/\hat{d}_G(u,w)) = O(d_G(u,v)/d_G(u,w))$, and the penalty is $O(\hat{d}_G(u,w)) = O(d_G(u,w))$. As a result, the expected stretch asymptotically remains unchanged. ◀

4.4 Efficient construction of approximate dominance sequences

Assume that $\hat{d}_G(u,v)$ is computed as $d_u(v)$ by the shortest-path algorithm in Section 3 from the source node u. Notice that $d_u(v)$ does not necessarily to be the same as $d_v(u)$, so $(G, \hat{d}_G(u,v))$ is not a metric space. Since the computed distances are distance preserving, it is easy to check that Lemma 11 is satisfied, which indicate that we can generate optimal tree embeddings based on the distances. This leads to the main theorem of this paper.

▶ **Theorem 12** (Efficient optimal tree embeddings). *There is a randomized algorithm that takes an undirected positively weighted graph $G = (V, E)$ containing $n = |V|$ vertices and $m = |E|$ edges, and produces an tree embedding such that for all $u, v \in V$, $d_G(u,v) \leq d_T(u,v)$ and $\mathbf{E}[d_T(u,v)] \leq O(\log n) \cdot d_G(u,v)$. The algorithm w.h.p. runs in $O(m \log n)$ time.*

The algorithm computes approximate dominance sequences $\hat{\chi}_\pi$ by the approximate SSSP algorithm introduced in Section 3. Then we apply Lemma 10 to convert $\hat{\chi}_\pi$ to an tree embedding. Notice that this tree embedding is an FRT-embedding based on $\hat{d}_G$. We still call this an FRT-embedding since the overall framework to generate hierarchical tree structure is similar to that in the original paper [17].

The advantage of our new SSSP algorithm is that the Decrease-Key and Extract-Min operation only takes constant time when the relative edge range (maximum divided by minimum) is no more than $n^{O(1)}$. To handle arbitrary weight edges we adopt a similar idea to [23] to solve the subproblems on specific edge ranges, and concatenate the results to form the final output. This requires pre-processing to restrict the edge range.

The high-level idea of the algorithm is as follows. In the pre-processing, the goal is to use $O(m \log n)$ time to generate a list of subproblems: the i-th subproblem has edge range in the interval $[n^{i-1}, n^{i+1}]$ and we compute the elements in the dominance sequences with values falling into the range from n^i to n^{i+1}. All the edge weights less than the minimum value of this range are treated as 0. Namely, the vertices form some components in one subproblem and the vertex distances within each component is 0.

After the subproblems are computed, we run the shortest-path algorithm on each subproblem, and the i-th subproblem generates the entries in the approximate dominance sequences in the range of $[n^i, n^{i+1})$. Finally, we solve an extra subproblem that only contains edges with weight less than n to generate the elements in dominance sequences whose distances fall in range $[1, n)$.

Due to page limit, the details of the algorithm, the proofs of the correctness and time bound are given in the full version of this paper.

5 An Application of FRT-Embedding: Ramsey Partitions and Distance Oracles

In this section we show a new application of FRT-embedding, which with our efficient construction, accelerates the construction of some existing approximate distance oracles [34, 10]. The bridge is to show that the FRT trees are Ramsey partitions [24]. It is interesting to point out that, the construction of FRT trees is not only much faster and simpler than the previous best-known approach [25] to generate Ramsey partitions, but the stretch factor of k, which is 18.5, is also smaller than the previous constants of 128 [24] and 33 [28].

We start with the definition of Ramsey partitions. Let (X, d_X) be a metric space. A hierarchical partition tree of X is a sequence of partitions $\{\mathcal{P}_k\}_{k=0}^{\infty}$ of X such that $\mathcal{P}_0 = \{X\}$, the diameter of the partitions in each level decreases by a constant $c > 1$, and each level $\mathcal{P}_{k+1}$ is a refinement of the previous level $\mathcal{P}_k$. A Ramsey partition [24] is a distribution of hierarchical partition trees such that each vertex has a lower-bounded probability of being sufficiently far from the partition boundaries in all partitions k, and this gap is called the *padded range* of a vertex. More formally:

▶ **Definition 13.** An (α, γ)-Ramsey partition of a metric space (X, d_X) is a probability distribution over hierarchical partition trees $\mathcal{P}$ of X such that for every $x \in X$:

$$\Pr\left[\forall k \in \mathbb{N}, B_X\left(x, \alpha \cdot c^{-k}\Delta\right) \subseteq \mathcal{P}_k(x)\right] \geq |X|^{-\gamma}.$$

An asymptotically tight construction of Ramsey partition where $\alpha = \Omega(\gamma)$ is provided by Mendel and Naor [24] using the Calinescu-Karloff-Rabani partition [9] for each level.

▶ **Theorem 14.** *The probability distribution over FRT trees is an asymptotically tight Ramsey Partition with $\alpha = \Omega(\gamma)$ (shown in the appendix) with fixed $c = 2$. More precisely, for every $x \in X$,*

$$\Pr\left[\forall i \in \mathbb{N}, B_X\left(x, \left(1 - 2^{-1/2a}\right)2^{-i}\Delta\right) \subseteq \mathcal{P}_i(x)\right] \geq \frac{1}{2}|X|^{-\frac{2}{a}}$$

for any positive integer $a > 1$.

The details of the proof are provided in the full version of this paper.

Ramsey Partitions are used in generating approximate distance oracles (ADOs), which supports efficient approximate pairwise shortest-distance queries. ADOs are well-studied by many researchers (e.g. [33, 5, 1, 16, 11, 24, 34, 10]), and a (P, S, Q, D)-distance oracle on a finite metric space (X, d_X) is a data structure that takes expected time P to preprocess from the given metric space, uses S storage space, and answers distance query between points x and y in X in time Q satisfying $d_X(x, y) \leq d_O(x, y) \leq D \cdot d_X(x, y)$, where $d_O(x, y)$ is the pairwise distance provided by the distance oracle, and D is called the stretch.

▶ **Corollary 15.** *With Theorem 14, we can accelerate the time to construct the Ramsey-partitions-based Approximate Distance Oracle in [24] to*

$$O\left(n^{1/k}(m + n\log n)\log n\right)$$

on a graph with n vertices and m edges, improving the stretch to $18.5k$, while maintaining the same storage space and constant query time.

This can be achieved by replacing the original hierarchical partition trees in the distance oracles by FRT trees (and some other trivial changes). The construction time can further reduce to $O\big(n^{1/k}m\log n\big)$ using the algorithm introduced in Section 4.4 while the oracle still has a constant stretch factor. Accordingly, the complexity to construct Christian Wulff-Nilsen's Distance Oracles [34] and Shiri Chechik's Distance Oracles [10] can be reduced to

$$O\Big(kmn^{1/k} + kn^{1+1/k}\log n + n^{1/ck}m\log n\Big)$$

since they all use Mendel and Naor's Distance Oracle to obtain an initial distance estimation. The acceleration is from two places: first, the FRT tree construction is faster; second, FRT trees provide better approximation bound, so the c in the exponent becomes smaller.

References

1. Rachit Agarwal and Philip Godfrey. Distance oracles for stretch less than 2. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 526–538, 2013.
2. Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995.
3. Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of IEEE Foundations of Computer Science (FOCS)*, pages 184–193, 1996.
4. Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 161–168. ACM, 1998.
5. Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 591–602, 2006.
6. Guy E. Blelloch, Yan Gu, Julian Shun, and Yihan Sun. Parallelism in randomized incremental algorithms. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 467–478, 2016.
7. Guy E. Blelloch, Yan Gu, and Yihan Sun. Efficient construction of probabilistic tree embeddings. *arXiv preprint arXiv:1605.04651*, 2016. URL: `https://arxiv.org/abs/1605.04651`.
8. Guy E. Blelloch, Anupam Gupta, and Kanat Tangwongsan. Parallel probabilistic tree embeddings, k-median, and buy-at-bulk network design. In *Proceedings of ACM symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 205–213, 2012.
9. Gruia Calinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, 34(2):358–372, 2005.
10. Shiri Chechik. Approximate distance oracles with constant query time. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 654–663, 2014.
11. Shiri Chechik. Approximate distance oracles with improved bounds. In *Proceedings of ACM on Symposium on Theory of Computing (STOC)*, pages 1–10, 2015.
12. Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, 1997.
13. Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM (JACM)*, 47(1):132–166, 2000.
14. Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m\log^{1/2} n$ time. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 343–352, 2014.
15. Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

16 Michael Elkin and Seth Pettie. A linear-size logarithmic stretch path-reporting distance oracle for general graphs. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 805–821, 2015.

17 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences (JCSS)*, 69(3):485–497, 2004.

18 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

19 Stephan Friedrichs and Christoph Lenzen. Parallel Metric Tree Embedding Based on an Algebraic View on Moore-Bellman-Ford. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 455–466, 2016.

20 Harold N. Gabow. Scaling algorithms for network problems. *Journal of Computer and System Sciences*, 31(2):148–168, 1985.

21 Mohsen Ghaffari and Christoph Lenzen. Near-optimal distributed tree embedding. In *International Symposium on Distributed Computing*, pages 197–211. Springer, 2014.

22 Maleq Khan, Fabian Kuhn, Dahlia Malkhi, Gopal Pandurangan, and Kunal Talwar. Efficient distributed approximation algorithms via probabilistic tree embeddings. *Distributed Computing*, 25(3):189–205, 2012.

23 Philip N. Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms*, 25(2):205–220, 1997.

24 Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 109–118, 2006.

25 Manor Mendel and Chaya Schwob. Fast C-K-R partitions of sparse graphs. *Chicago Journal of Theoretical Computer Science*, pages 1–18, 2009. Article 2. `doi:10.4086/cjtcs.2009.002`.

26 Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 192–201, 2015.

27 Donald R. Morrison. PATRICIA – Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM (JACM)*, 15(4):514–534, October 1968.

28 Assaf Naor and Terence Tao. Scale-oblivious metric fragmentation and the nonlinear Dvoretzky theorem. *Israel Journal of Mathematics*, 192(1):489–504, 2012.

29 Seth Pettie and Vijaya Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

30 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 255–264, 2008.

31 Raimund Seidel. *Backwards analysis of randomized geometric algorithms*. Springer, 1993.

32 Mikkel Thorup. Undirected single source shortest paths in linear time. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 12–21, 1997.

33 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

34 Christian Wulff-Nilsen. Approximate distance oracles with improved query time. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 539–549, 2013.

Approximating Partition Functions of Bounded-Degree Boolean Counting Constraint Satisfaction Problems*†

Andreas Galanis[1], Leslie Ann Goldberg[2], and Kuan Yang[3]

1 **University of Oxford, Oxford, UK**
andreas.galanis@cs.ox.ac.uk
2 **University of Oxford, Oxford, UK**
leslie.goldberg@cs.ox.ac.uk
3 **University of Oxford, Oxford, UK**
kuan.yang@cs.ox.ac.uk

Abstract

We study the complexity of approximate counting Constraint Satisfaction Problems (#CSPs) in a bounded degree setting. Specifically, given a Boolean constraint language Γ and a degree bound Δ, we study the complexity of $\#\mathsf{CSP}_\Delta(\Gamma)$, which is the problem of counting satisfying assignments to CSP instances with constraints from Γ and whose variables can appear at most Δ times. Our main result shows that: (i) if every function in Γ is affine, then $\#\mathsf{CSP}_\Delta(\Gamma)$ is in FP for all Δ, (ii) otherwise, if every function in Γ is in a class called IM_2, then for all sufficiently large Δ, $\#\mathsf{CSP}_\Delta(\Gamma)$ is equivalent under approximation-preserving (AP) reductions to the counting problem $\#\mathsf{BIS}$ (the problem of counting independent sets in bipartite graphs) (iii) otherwise, for all sufficiently large Δ, it is NP-hard to approximate the number of satisfying assignments of an instance of $\#\mathsf{CSP}_\Delta(\Gamma)$, even within an exponential factor. Our result extends previous results, which apply only in the so-called "conservative" case.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures

Keywords and phrases Constraint Satisfaction, Approximate Counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.27

1 Introduction

Constraint Satisfaction Problems (CSPs), which originated in Artificial Intelligence [18] provide a general framework for modelling decision, counting and approximate counting problems. The paradigm is sufficiently general that applications from diverse areas such as database theory, scheduling and graph theory can all be captured (see, for example, [14, 15, 17]). Moreover, all graph homomorphism decision and counting problems [12] can be re-cast in the CSP framework and partition function problems from statistical physics [21] can be represented as counting CSPs. Given the usefulness of CSPs, the study of the complexity

* A full version of the paper (with the same theorem-numbering) containing detailed proofs is available at http://arxiv.org/abs/1610.04055.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein. Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK.

of CSPs is a an extremely active area in computational complexity (for example, see [2] and the references therein).

In this paper, we are concerned with Boolean counting CSPs. An instance $I = (V, \mathcal{C})$ of a Boolean counting CSP consists of a set V of *variables* and a set $\mathcal{C}$ of constraints. An assignment $\sigma : V \to \{0, 1\}$ assigns a Boolean value called a "spin" to each variable. Each constraint associates a tuple $(v_1, \ldots, v_k)$ of variables with a Boolean relation which constrains the spins that can be assigned to $v_1, \ldots, v_k$. The assignment σ is said to "satisfy" the constraint if the tuple $(\sigma(v_1), \ldots, \sigma(v_k))$ is in the corresponding relation. An assignment is said to be "satisfying" if it satisfies all constraints. A Constraint Satisfaction Problem comes with two important parameters – the constraint language Γ is the set of all relations that may be used in constraints and the degree Δ is the maximum number of times that any variable $v \in V$ may be used in constraints in any instance. The number of satisfying assignments is denoted Z_I. The computational problem $\#\mathsf{CSP}_\Delta(\Gamma)$ is the problem of computing Z_I, given a CSP instance I with constraints in Γ and degree at most Δ. We use $\#\mathsf{CSP}(\Gamma)$ to denote the version of the problem in which the degree of instances is unconstrained.

Although constraints are supported by Boolean relations, they can be used to code up weighted interactions such as those that arise in statistical physics. For example, let R be the "not-all-equal" relation of arity 3. Then consider the conjunction of $R(x, a, b)$ and $R(y, a, b)$. There are two satisfying assignments with $\sigma(x) = 0$ and $\sigma(y) = 1$ since $\sigma(a)$ and $\sigma(b)$ must differ. Similarly, there are two satisfying assignments with $\sigma(x) = 1$ and $\sigma(y) = 0$. On the other hand, there are three satisfying assignments with $\sigma(x) = \sigma(y) = 1$ and there are three satisfying assignments with $\sigma(x) = \sigma(y) = 0$. Thus, the induced interaction on the variables x and y is the same as the interaction of the ferromagnetic Ising model (at an appropriate temperature) – an assignment in which x and y have the same spin has weight 3, whereas an assignment where they have different spins has weight 2.

For every $\Delta \geq 3$, the work of Cai, Lu and Xia [5] completely classifies the complexity of exactly solving $\#\mathsf{CSP}_\Delta(\Gamma)$, depending on the parameter Γ. If every relation in Γ is affine, then $\#\mathsf{CSP}_\Delta(\Gamma)$ is solvable in polynomial time (so the problem in the complexity class FP). Otherwise, it is $\#\mathsf{P}$-complete. The term "affine" will be defined in § 2. Roughly, it means that the tuples in the relation are solutions to a linear system, so Gaussian elimination gives an appropriate polynomial-time algorithm. The characterisation of Cai, Lu and Xia is exactly the same classification that was obtained for the unbounded problem $\#\mathsf{CSP}(\Gamma)$ by Creignou and Hermann [6]. Thus, as far as exact counting is concerned, the degree-bound Δ does not affect the complexity as long as $\Delta \geq 3$. As Cai, Lu and Xia point out, the dichotomy is false for $\Delta = 2$, where $\#\mathsf{CSP}_2(\Gamma)$ is equivalent to the Holant problem $\mathsf{Holant}(\Gamma)$ – see the references in [5] for more information about Holant problems.

Much less is known about the complexity of *approximately* solving $\#\mathsf{CSP}_\Delta(\Gamma)$. In fact, even the *decision problem* is still open. While Schaefer [19] completely classified the complexity of the decision problem $\mathsf{CSP}(\Gamma)$ – where the goal is to determine whether or not Z_I is 0 for an instance of $\#\mathsf{CSP}(\Gamma)$ – the complexity of the corresponding decision problem $\mathsf{CSP}_\Delta(\Gamma)$, where the instance has degree at most Δ, is still not completely resolved. For $\Delta \geq 3$, Dalmau and Ford [7] have solved the special case where Γ includes both of the relations $R_{\delta_0} = \{0\}$ and $R_{\delta_1} = \{1\}$. This special case is known as the "conservative case" in the CSP literature. For $\Delta \geq 6$, Dyer et al. [9] have classified the difficulty of the approximation problem:

- If every relation in Γ is affine, then $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ is in FP.
- Otherwise, if every relation in Γ is in a class called IM_2 (a class defined in § 2) then $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ is equivalent under approximation-preserving (AP) reductions to the counting problem $\#\mathsf{BIS}$ (the problem of counting independent sets in bipartite graphs).
- Otherwise, there is no FPRAS for $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ unless $\mathsf{NP} = \mathsf{RP}$.

Dyer et al. made only partial progress on the cases where $\Delta \in \{3, 4, 5\}$. We refer the reader to [9, 16] for a discussion of the partial classification. However, it is worth noting here that the complexity of $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ is closely related to the complexity of counting satisfying assignments of so-called read-d Monotone CNF Formulas. Crucial progress was made by Liu and Lu [16], who completely resolved the complexity of the latter problem. Given the work of Liu and Lu, a complete classification of $\#\mathsf{CSP}_\Delta(\Gamma \cup \{R_{\delta_0}, R_{\delta_1}\})$ for $\Delta \in \{3, 4, 5\}$ may be in reach.

The restriction that R_{δ_0} and R_{δ_1} are contained in Γ is a severe one because it does not apply to many natural applications. On the other hand, we are a long way from a precise understanding of the complexity of $\#\mathsf{CSP}_\Delta(\Gamma)$ without this restriction because there are specific, relevant parameters that we do not understand. For example, for a positive integer k, let Γ be the singleton set containing only the arity-k "not-all-spin-1" relation. Then satisfying assignments of an instance of $\#\mathsf{CSP}_\Delta(\Gamma)$ correspond to independent sets of a k-uniform hypergraph with maximum degree Δ. It is known that there is an FPRAS for $\Delta = O(2^{k/2})$ [13] and that the problem is NP-hard to approximate for $\Delta = \Omega(2^{k/2})$ [1]; the implicit constants in these bounds do not currently match and thus, for big k, there is a large range of Δ's where we do not yet know the complexity of approximating $\#\mathsf{CSP}_\Delta(\Gamma)$. If Γ instead contains (only) the arity-k "at-least-one-spin-0" relation then satisfying assignments of an instance of $\#\mathsf{CSP}_\Delta(\Gamma)$ correspond to the so-called "strong" independent sets of a k-uniform hypergraph. Song, Yin and Zhao [20] have presented a barrier for hardness results, showing why current technology is unsuitable for resolving the cases where $\Delta \in \{4, 5\}$ (roughly, these cases are in "non-uniqueness", but this is not realisable by finite gadgets).

The purpose of the present paper is to remove the severe restriction that R_{δ_0} and R_{δ_1} are contained in Γ in the approximate counting classification of $\#\mathsf{CSP}_\Delta(\Gamma)$ from [9]. Since pinning down precise thresholds seems a long way out of reach, we instead focus on whether there is a "barrier" value Δ_0 such that, for all $\Delta \geq \Delta_0$, approximation is intractable. Since we wish to get the strongest possible inapproximability results (showing the hardness of approximating Z_I even within an exponential factor), we define the following computational problem, which has an extra parameter $c > 1$ that captures the desired accuracy.

Name $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$.
Instance An n-variable instance I of a CSP with constraint language Γ and degree at most Δ.
Output A number $\widehat{Z}$ such that $c^{-n} Z_I \leq \widehat{Z} \leq c^n Z_I$.

Although we have not yet defined all of the terms, we can now at least state (a weak version of) our result.

▶ **Theorem 1.** *Let Γ be a Boolean constraint language. Then,*
1. *If every relation in Γ is affine then $\#\mathsf{CSP}(\Gamma)$ is in FP.*
2. *Otherwise, if every relation in Γ is in the class IM_2, then there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, $\#\mathsf{CSP}_\Delta(\Gamma)$ is $\#\mathsf{BIS}$-equivalent under AP-reductions.*
3. *Otherwise, there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, there exists a real number $c > 1$ such that $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ is NP-hard.*

After defining all of the terms, we will state a stronger theorem, Theorem 6, which immediately implies Theorem 1. The stronger version applies to the #CSP problems that we have already introduced, but it also applies to other restrictions of these problems, which have even more applications.

We now explain the restriction. Note that in the CSP framework, as we have defined it, the variables that are constrained by a given constraint need not be distinct. Thus, if the arity-4 relation R is present in a constraint language Γ, then an instance of $\#\mathsf{CSP}(\Gamma)$

with variables x and y may contain a constraint such as $R(x, x, y, x)$. This ability to repeat variables is equivalent to assuming that equality relations of all arities are present in Γ. This feature of the CSP definition is inconvenient for two reasons: (1) It does not fit well with some spin-system applications, and (2) In many settings, it obscures the nuanced complexity classification that arise.

As an example of (1), recall the application where Γ is the singleton set containing only the arity-k "not-all-spin-1" relation. As we noted earlier, satisfying assignments of a $\#\mathsf{CSP}(\Gamma)$ instance correspond to independent sets of a k-uniform hypergraph. Here, hyperedges are size-k subsets of vertices and it does not make sense to allow repeated vertices!

The point (2) is well-known. In fact, the "equality is always present" assumption is the main feature that separates #CSPs from the more general Holant framework [3].

In our current setting, it turns out that adding equality functions to Γ does not change the complexity classification, but this is a result of our theorems rather than an a priori assumption – indeed, determining which constraint languages Γ can appropriately simulate equality functions is one of the difficulties – thus, throwing equalities in "for free" would substantially weaken our results! Our main result, Theorem 6, which will be presented in § 2, applies both to the #CSPs that we have already defined, and to more refined versions, in which constraints may not repeat variables.

We wish now to discuss an important special case in which both the #CSPs and the refined versions have already been studied. This is the special case in which Γ consists of a single relation which is symmetric in its arguments. A symmetric relation that is not affine is not in IM_2. Therefore, Item 2 in the statement of Theorem 1 never arises in this special case. Our earlier paper [11] shows that, in this case (where Γ consists of a single, symmetric, non-affine relation) there is an integer Δ_0 such that for all $\Delta \geq \Delta_0$, there exists a real number $c > 1$ such that $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ is NP-hard.

While the work of [11] is important for this paper, note that the special case is far from general – in particular, it is easy to induce asymmetric constraints using symmetric ones. For example, suppose that R_1 is the (symmetric) arity-2 "not-all-spin-1" constraint, R_2 is the (symmetric) arity-2 "not the same spin" constraint and $R_3 = \{(0,0),(0,1),(1,1)\}$ is the (asymmetric) arity-2 "Implies" constraint. Then the conjunction of $R_1(x, a)$ and $R_2(a, y)$ induces $R_3(x, y)$.

It is interesting that Theorem 1 is exactly the same classification that was obtained for the *unbounded* problem $\#\mathsf{CSP}(\Gamma)$ by Dyer et al. [10]. In particular, they showed

1. If every relation in Γ is affine then $\#\mathsf{CSP}(\Gamma)$ is in FP.
2. Otherwise, if every relation in Γ is in the class IM_2, then $\#\mathsf{CSP}(\Gamma)$ is $\#\mathsf{BIS}$-equivalent under AP-reductions.
3. Otherwise, $\#\mathsf{CSP}(\Gamma)$ is $\#\mathsf{SAT}$-equivalent under AP-reductions, where $\#\mathsf{SAT}$ is the problem of counting the satisfying assignments of a Boolean formula.

The inapproximability that we demonstrate in Item 3 of Theorem 1 is stronger than what was known in the unbounded case, both (obviously) because of the degree bound, but also because we show that it is hard to get within an exponential factor. (This strong kind of inapproximability was also missing from the results of [9]).

2 Definitions and Statement of Main Result

Before giving formal definitions of the problems that we study, we introduce some notation. We use boldface letters to denote Boolean vectors. A *pseudo-Boolean* function is a function of the form $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ for some positive integer k, which is called the *arity* of f.

▶ **Definition 2.** Given a pseudo-Boolean function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$, we use the notation R_f to denote the relation $R_f = \{\mathbf{x} \in \{0,1\}^k \mid f(\mathbf{x}) > 0\}$, which is the relation underlying f.

If the range of f is $\{0,1\}$ then f is said to be a *Boolean function* and of course in that case $R_f = \{\mathbf{x} \in \{0,1\}^k \mid f(\mathbf{x}) = 1\}$.

In order to allow consistency with obvious generalisations, our formal definition of the Boolean Constraint Satisfaction Problem is in terms of Boolean functions (rather than, equivalently, using the underlying relations).

A *Constraint language* Γ is a set of pseudo-Boolean functions. It is a *Boolean constraint language* if all of the functions in it are Boolean functions. An instance $I = (V, \mathcal{C})$ of a CSP with constraint language Γ consists of a set V of variables and a set $\mathcal{C}$ of constraints. Each constraint $C_i \in \mathcal{C}$ is of the form $f_i(v_{i,1}, \ldots, v_{i,k_i})$ where f_i is an arity-k_i function in Γ and $(v_{i,1}, \ldots, v_{i,k_i})$ is a tuple of (not necessarily distinct) variables in V. The constraint C_i is said to be "Repeat-Free" if all of the variables are distinct. Each *assignment* $\sigma : V \to \{0,1\}$ of Boolean values to the variables in V has a weight $w_I(\sigma) := \prod_{f_i(v_{i,1},\ldots,v_{i,k_i}) \in \mathcal{C}} f_i(\sigma(v_{i,1}), \ldots, \sigma(v_{i,k_i}))$. The *partition function* maps the instance I to the quantity $Z_I := \sum_{\sigma:V\to\{0,1\}} w_I(\sigma) = \sum_{\sigma:V\to\{0,1\}} \prod_{f_i(v_{i,1},\ldots,v_{i,k_i}) \in \mathcal{C}} f_i(\sigma(v_{i,1}), \ldots, \sigma(v_{i,k_i}))$.

If Γ is a Boolean constraint language then it is easy to see that $w_I(\sigma) = 1$ if the assignment is satisfying and $w_I(\sigma) = 0$, otherwise. Thus, Z_I is the number of satisfying assignments of I.

When $Z_I > 0$, we will use $\mu_I(\cdot)$ to denote the Gibbs distribution corresponding to Z_I. This is the probability distribution on the set of assignments $\sigma : V \to \{0,1\}$ such that $\mu_I(\sigma) = w_I(\sigma)/Z_I$ for all $\sigma : V \to \{0,1\}$.

The *degree* $d_v(C)$ of a variable v in a constraint C is the number of times that the variable v appears in the tuple corresponding to C and the degree d_v of the variable is $d_v = \sum_{C\in\mathcal{C}} d_v(C)$. Finally, the degree of the instance I is $\max_{v\in V} d_v$.

▶ **Definition 3.** $\#\mathsf{CSP}_\Delta(\Gamma)$ is the problem of computing Z_I, given a CSP instance I with constraints in Γ and degree at most Δ. $\#\mathsf{CSP}(\Gamma)$ is the version of the problem in which the degree of instances is unconstrained. $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ has an extra parameter $c > 1$ that captures the desired accuracy. The problem is to compute a number $\widehat{Z}$ such that $c^{-n} Z_I \leq \widehat{Z} \leq c^n Z_I$, where n is the number of variables in the instance I. The problems $\#\mathsf{NoRepeatCSP}_\Delta(\Gamma)$, $\#\mathsf{NoRepeatCSP}(\Gamma)$ and $\#\mathsf{NoRepeatCSP}_{\Delta,c}(\Gamma)$ are defined similarly, except that inputs are restricted so that all constraints are Repeat-Free.

▶ **Definition 4.** A Boolean function $f : \{0,1\}^k \to \{0,1\}$ is *affine* if there is a $k \times k$ Boolean matrix $\mathbf{A}$ and a length-k Boolean vector $\mathbf{b}$ such that R_f is equal to the set of solutions $\mathbf{x}$ of $\mathbf{Ax} = \mathbf{b}$ over GF(2).

▶ **Definition 5** (The set of functions IM_2)**.** A Boolean function $f : \{0,1\}^k \to \{0,1\}$ is in IM_2 if $f(x_1, \ldots, x_k)$ is logically equivalent to a conjuction of (any number of) predicates of the form x_i, $\neg x_i$ or $x_i \Rightarrow x_j$.

We have now defined all of the terms in our main theorem apart from some well-known concepts from complexity theory, which we discuss next. FP is the class of computational problems (with numerical output) that can be solved in polynomial time. An FPRAS is a randomised algorithm that produces approximate solutions within specified relative error with high probability in polynomial time. For two counting problems #A and #B, we say that #A is #B-easy if there is an approximation-preserving (AP)-reduction from #A to #B. The formal definition of an AP-reduction can be found in [8]. It is a randomised Turing reduction that yields close approximations to #A when provided with close approximations to #B. The definition of AP-reduction meshes with the definition of FPRAS in the sense

that the existence of an FPRAS for #B implies the existence of an FPRAS for #A. We say that #A is #B-hard if there is an AP-reduction from #B to #A. Finally, we say that #A is #B-equivalent if #A is both #B-easy and #B-hard.

The problem of counting satisfying assignments of a Boolean formula is denoted by #SAT. Every counting problem in #P is AP-reducible to #SAT, so #SAT is said to be complete for #P with respect to AP-reductions. It is known that there is no FPRAS for #SAT unless RP = NP. The problem of counting independent sets in a bipartite graph is denoted by #BIS. The problem #BIS appears to be of intermediate complexity: there is no known FPRAS for #BIS (and it is generally believed that none exists) but there is no known AP-reduction from #SAT to #BIS. Indeed, #BIS is complete with respect to AP-reductions for a complexity class $\#\mathsf{RH}\Pi_1$.

Given all of these definitions, we now formally state the stronger version of Theorem 1 promised in the introduction. The proof can be found in full version.

▶ **Theorem 6.** *Let Γ be a Boolean constraint language. Then,*

1. *If every function in Γ is affine then $\#\mathsf{CSP}(\Gamma)$ and $\#\mathsf{NoRepeatCSP}(\Gamma)$ are both in FP.*
2. *Otherwise, if $\Gamma \subseteq IM_2$, then there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, $\#\mathsf{CSP}_\Delta(\Gamma)$ and $\#\mathsf{NoRepeatCSP}_\Delta(\Gamma)$ are both #BIS-equivalent under AP-reductions, and*
3. *Otherwise, there exists an integer Δ_0 such that for all $\Delta \geq \Delta_0$, there exists a real number $c > 1$ such that $\#\mathsf{CSP}_{\Delta,c}(\Gamma)$ and $\#\mathsf{NoRepeatCSP}_{\Delta,c}(\Gamma)$ are both NP-hard.*

3 Overview of the Proof of Theorem 6

In this section, we give a non-technical overview of the proof of Theorem 6. Our objective is to illustrate the main ideas and obstacles without delving into the more detailed definitions. A more technical overview can be found in § 5. Our focus in this section will be on the case where Γ consists of a single Boolean function $f : \{0,1\}^k \to \{0,1\}$. This case is the main ingredient in the proof of the theorem.

A typical approach for showing that a counting CSP is intractable is to use an instance of the CSP to build a "gadget" which simulates an intractable binary 2-spin constraint. This was the approach used in [11], which proved the intractability of $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ for any *symmetric* non-affine Boolean function f. There, an instance I of $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ was constructed, along with variables x and y, such that for all spins $s_x \in \{0,1\}$ and $s_y \in \{0,1\}$ the marginal distribution $\mu_I(x,y)$ satisfies

$$\mu_I(\sigma(x) = s_x, \sigma(y) = s_y) = g(s_x, s_y)/(g(0,0) + g(0,1) + g(1,0) + g(1,1)), \tag{1}$$

where g is a binary function that codes up the interaction of an intractable anti-ferromagnetic 2-spin system. We will not need to give detailed definitions of 2-spin systems in this paper. Instead, we give a sufficient condition for intractability.

▶ **Definition 7.** A binary function $g : \{0,1\}^2 \to \mathbb{R}_{\geq 0}$ is said to be "*hard*" if all of the following hold: $g(0,0)+g(1,1) > 0$, $\min\{g(0,0), g(1,1)\} < \sqrt{g(0,1)g(1,0)}$, and $\max\{g(0,0), g(1,1)\} \leq \sqrt{g(0,1)g(1,0)}$.

It was established in [11] that the ability to "simulate" a hard function g in the sense of (1) ensures that $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ is NP-hard to approximate, even within an exponential factor.

A key feature of *symmetric* Boolean functions f which facilitated such simulation in [11] was the fact that the class of relevant hard functions g is well-behaved, and it turned out that

it suffices to encode such a hard binary function with only ϵ-accuracy, for some sufficiently small $\epsilon > 0$, and this was enough to ensure the NP-hardness of $\#\mathsf{CSP}_{\Delta,c}(\{f\})$.

The main obstacle in adapting the approach of [11] to the case where f need not be symmetric in its arguments arises when f is in IM_2. It is unlikely that such a function f can simulate a hard function g in the sense of (1) – indeed such a simulation would prove the (very surprising) result that #BIS does not have an FPRAS (unless NP = RP). Thus, for $f \in IM_2$, we need instead to encode a binary function which will allow us to connect the problem $\#\mathsf{NoRepeatCSP}_{\Delta}(\{f\})$ to #BIS.

Now consider the binary Boolean function Implies whose underlying relation $R_{\mathsf{Implies}} = \{(0,0),(0,1),(1,1)\}$ contains all (x,y) satisfying $x \Rightarrow y$. Obviously, Implies is not symmetric, and it is not hard according to Definition 7. On bipartite instances, however, the symmetry can be restored by interpreting differently the spins 0 and 1 on the two parts of the graph, and this leads to a connection with #BIS. In particular, it is well-known [10] that $\#\mathsf{CSP}(\{\mathsf{Implies}\})$ is equivalent to #BIS under AP-reductions. This connection was extended to the bounded-degree setting by [4].

Unfortunately, the symmetrisation which connects $\#\mathsf{CSP}(\{\mathsf{Implies}\})$ to #BIS is not very robust. For example, suppose that a (non-symmetric) Boolean function f can be used to simulate, in the sense of (1), a binary function g which is very close to Implies. In particular, suppose that for some $\epsilon > 0$ and $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ satisfying $|\epsilon_i| \leq \epsilon$ for $i = 1, 2, 3, 4$, we have $g(0,0) = 1+\epsilon_1$, $g(0,1) = 1+\epsilon_2$, $g(1,0) = \epsilon_3$, and $g(1,1) = 1+\epsilon_4$. Such a close approximation is about the best that can be expected using the kind of approximate encodings that are available. However, the complexity of asymmetric 2-spin systems is not sufficiently well understood to exploit such a simulation. Surprisingly, for *any* arbitrarily small constant $\epsilon > 0$, it is not known even whether the unbounded degree version $\#\mathsf{CSP}(\{g\})$ is #BIS-hard, and certainly nothing is known in our bounded-degree setting! The trouble is that the symmetrisation that works for Implies (i.e., when $\epsilon_i = 0$ for $i = 1, 2, 3, 4$) is no longer guaranteed to symmetrise the imperfect version with the ϵ_i's, so the swapping of spin-0 and spin-1 values on one side of the bipartite graph leads to an *asymmetric* 2-spin system on bipartite graphs and this does not fall into the scope of known results [4] concerning bounded-degree bipartite 2-spin systems.

Our approach to handle this problem for $f \in IM_2$ is to carefully ensure that there is no accuracy error ϵ in encoding the function Implies. In other words, we show that, using $f \in IM_2$, we can encode Implies *perfectly*, a task which is surprisingly intricate in the repeat-free setting. Our main technical theorem, Theorem 17, achieves this goal. Namely, it shows that, for every non-affine Boolean function f, either f simulates a hard function (with arbitrarily small accuracy-error ϵ, which leads to the desired intractability of $\#\mathsf{NoRepeatCSP}_{\Delta}(\{f\})$) or else f "supports perfect equality" – a concept which will be defined later, but essentially means that f can be used to perfectly simulate the binary function EQ with underlying relation $R_{\mathsf{EQ}} = \{(0,0),(1,1)\}$. Using EQ, it is possible to simulate repeated variables in constraints, so the #BIS-hardness of $\#\mathsf{CSP}_{\Delta}(\{f\})$ follows from [10]. When $f \notin IM_2$ but f supports perfect equality, instead of reducing to the work in [10], we work somewhat harder to make sure that we also get the strong (exponential factor) inapproximability given in Theorem 6.

4 Pinning, equality and simulating functions

An important case in our proof is the case where Γ contains a single function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. In this case, we can we simplify the notation because the constraints in an instance I are in one-to-one correspondence with k-tuples of variables (there is no need to repeat the name of the function f in each constraint). So, for convenience, we make the following definitions.

A *k-tuple hypergraph* $H = (V, \mathcal{F})$ consists of a set V of vertices, together with a set $\mathcal{F}$ of *hyperarcs*, where every hyperarc in $\mathcal{F}$ is a k-tuple of distinct vertices in V. The degree of H is the maximum, over all vertices $v \in V$, of the number of hyperarcs that contain v. Given a function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$, we let $I_f(H)$ denote the instance of $\#\mathsf{NoRepeatCSP}(\{f\})$ whose constraints correspond to the hyperarcs of H. Given an assignment $\sigma\colon V \to \{0,1\}$ we define $w_{f;H}(\sigma) := \prod_{(v_1,\ldots,v_k)\in\mathcal{F}} f(\sigma(v_1),\ldots,\sigma(v_k))$ and $Z_{f;H} := \sum_{\sigma:V\to\{0,1\}} w_{f;H}(\sigma)$, so $Z_{I_f(H)} = Z_{f;I_f(H)}$. By analogy to the Gibbs distribution on satisfying assignments, when $Z_{f;H} > 0$, we use $\mu_{f;H}(\cdot)$ to denote the probability distribution in which, for all assignments $\sigma : V \to \{0,1\}$, $\mu_{f;H}(\sigma) = w_{f;H}(\sigma)/Z_{f;H}$.

Given a function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and a positive integer Δ, we define $\#\mathsf{Multi2Spin}_\Delta(f)$ to be the problem of computing $Z_{f;H}$, given as input a k-tuple hypergraph H with degree at most Δ. The name $\#\mathsf{Multi2Spin}_\Delta(f)$ indicates that the problem is to compute the partition function of a 2-spin system with multi-body interactions specified by f and degree-bound Δ. Given a real number $c > 1$, the problem $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ has the same input, and the goal, when the input has n vertices, is to compute a number $\widehat{Z}$ such that $c^{-n}Z_{f;H} \leq \widehat{Z} \leq c^n Z_{f;H}$. Clearly, $\#\mathsf{Multi2Spin}_\Delta(f)$ is equivalent to $\#\mathsf{NoRepeatCSP}_\Delta(\{f\})$ and $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ is equivalent to $\#\mathsf{NoRepeatCSP}_{\Delta,c}(\{f\})$.

4.1 Supporting pinning and equality

Let k be a positive integer and let $H = (V, \mathcal{F})$ be a k-tuple hypergraph. Given a configuration $\sigma : V \to \{0,1\}$ and a subset $T \subseteq V$, we will use σ_T to denote the restriction of σ to vertices in T. For a vertex $v \in V$, we will also use σ_v to denote the spin $\sigma(v)$ of vertex v in σ. The following definitions are generalisations of definitions from [11].

▶ **Definition 8.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. Suppose that $\epsilon \geq 0$ and $s \in \{0,1\}$. The k-tuple hypergraph H is an *ϵ-realisation of pinning-to-s* if there exists a vertex v of H such that $\mu_{f;H}(\sigma_v = s) \geq 1 - \epsilon$.

▶ **Definition 9.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and $s \in \{0,1\}$. We say that f supports *pinning-to-s* if, for every $\epsilon > 0$, there is a k-tuple hypergraph which is an ϵ-realisation of pinning-to-s. We say that f supports *perfect* pinning-to-s if there is a k-tuple hypergraph which is a 0-realisation of pinning-to-s.

We now define what it means for a function f to support (perfect) equality (cf. § 3).

▶ **Definition 10.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and $\epsilon \geq 0$. The k-tuple hypergraph H is an *ϵ-realisation of equality* if there exist distinct vertices v_1 and v_2 of H such that, for each $s \in \{0,1\}$, $\mu_{f;H}(\sigma_{v_1} = \sigma_{v_2} = s) \geq (1-\epsilon)/2$.

▶ **Definition 11.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. The function f *supports equality* if, for every $\epsilon > 0$, there is a k-tuple hypergraph which is an ϵ-realisation of equality. The function f *supports perfect equality* if there is a k-tuple hypergraph which is a 0-realisation of equality.

4.2 Realising conditional distributions induced by pinning and equality

Given a set S of vertices, we write $\sigma_S = \mathbf{0}$ to denote the event that all vertices in S are assigned the spin 0 under the assignment σ. We similarly write $\sigma_S = \mathbf{1}$ to denote the event that all vertices in S are assigned the spin 1 under the assignment σ. Finally, we use use σ_S^{eq} to denote the event that all vertices in S have the same spin under σ (the spin could be 0 or 1). The following definition is a generalisation of Definition 16 of [11] except that we have changed the notation slightly for convenience.

▶ **Definition 12** ([11, Definition 16]). Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. Let $H = (V, \mathcal{F})$ be a k-tuple hypergraph. Let $\mathcal{V} = (V_{\mathsf{pin0}}, V_{\mathsf{pin1}}, \mathcal{V}_{\mathsf{eq}})$ where V_{pin0} and V_{pin1} are disjoint subsets of V and $\mathcal{V}_{\mathsf{eq}}$ is a (possibly empty) set of disjoint subsets of $V \backslash (V_{\mathsf{pin0}} \cup V_{\mathsf{pin1}})$. Suppose that: (i) $V_{\mathsf{pin0}} = \emptyset$ if f does not support pinning-to-0, (ii) $V_{\mathsf{pin1}} = \emptyset$ if f does not support pinning-to-1, (iii) $\mathcal{V}_{\mathsf{eq}} = \emptyset$ if f does not support equality, (iv) it holds that $\mu_{f;H}(\sigma_{V_{\mathsf{pin0}}} = \mathbf{0}, \sigma_{V_{\mathsf{pin1}}} = \mathbf{1}, \bigcap_{W \in \mathcal{V}_{\mathsf{eq}}} \sigma^{\mathsf{eq}}_W) > 0$. We will then say that "$\mathcal{V}$ is *admissible* for H with respect to f" and we will denote by $\mu^{\mathrm{cond}(\mathcal{V})}_{f;H}$ the probability distribution $\mu_{f;H}(\cdot \mid \sigma_{V_{\mathsf{pin0}}} = \mathbf{0}, \sigma_{V_{\mathsf{pin1}}} = \mathbf{1}, \bigcap_{W \in \mathcal{V}_{\mathsf{eq}}} \sigma^{\mathsf{eq}}_W)$.

4.3 Simulating hard functions and inapproximability results

We can now give a formal definition of "simulation", along the lines that was informally discussed in § 3 (Equation (1)).

▶ **Definition 14.** Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ and $g : \{0,1\}^t \to \mathbb{R}_{\geq 0}$. The function f simulates the function g if there is a k-tuple hypergraph H, an admissible set $\mathcal{V}$ for H with respect to f, and t vertices $v_1, v_2, \ldots, v_t$ of H such that, for all $(s_1, s_2, \ldots, s_t) \in \{0,1\}^t$,

$$\mu^{\mathrm{cond}(\mathcal{V})}_{f;H}(\sigma(v_1) = s_1, \sigma(v_2) = s_2, \ldots, \sigma(v_t) = s_t) = \frac{g(s_1, s_2, \ldots, s_t)}{\sum_{(s'_1, s'_2, \ldots, s'_t) \in \{0,1\}^t} g(s'_1, s'_2, \ldots, s'_t)}.$$

If $\mathcal{V} = (\emptyset, \emptyset, \emptyset)$, then we say that f *perfectly simulates* g. More generally, we say that f simulates a set of functions $\mathcal{G}$ if f simulates every $g \in \mathcal{G}$.

The connection betweeen "hard" as defined in Definition 7 and intractability is given in the following lemma. The lemma is stated for symmetric functions in [11], but the proof also works for asymmetric functions.

▶ **Lemma 15** ([11, Lemma 18]). *Let $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$. If f simulates a hard function, then for all sufficiently large Δ, there exists $c > 1$ such that $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ is NP-hard.* ◀

5 Proof Sketch

In this section, for a Boolean function $f : \{0,1\}^k \to \{0,1\}$, we consider the complexity of the problems $\#\mathsf{Multi2Spin}_{\Delta}(f)$ and $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$. Classifying the complexity of these problems is the most important step in the proof of Theorem 6. Namely, to obtain Theorem 6, it suffices to show that for every non-affine function f, we have that:

- If f is in IM_2, then for all sufficiently large Δ, $\#\mathsf{Multi2Spin}_{\Delta}(f)$ is $\#\mathsf{BIS}$-equivalent.
- If f is not in IM_2, then for all sufficiently large Δ, there exists a real number $c > 1$ such that $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ is NP-hard.

Our main technical theorem to prove this is the following classification of Boolean functions, which asserts that every non-affine function either supports perfect equality or simulates a hard function. A proof sketch is provided later.

▶ **Theorem 17.** *Let $k \geq 2$ and let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function. Then at least one of three following propositions is true:*

1. *f is affine;*
2. *f supports perfect equality;*
3. *f simulates a hard function.*

When f simulates a hard function, using Lemma 15, we can immediately conclude that for all sufficiently large Δ, there exists $c > 1$ such that $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ is NP-hard. As

we already discussed in § 3, it is important that, in the case where f does not simulate a hard function, Theorem 17 guarantees that f supports *perfect* equality (rather than simple imperfect equality); this allows us to recover the connection to #BIS for those $f \in IM_2$. In fact, when f supports perfect equality, we can effectively carry out (a strengthening of) the program in [10] to obtain the following classification which perfectly aligns with Theorem 6.

▶ **Theorem 18.** *Let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function that is not affine. Suppose that f supports perfect equality.*

1. *If f is in IM_2, then for all sufficiently large Δ, $\#\mathsf{Multi2Spin}_\Delta(f)$ is $\#\mathsf{BIS}$-equivalent.*
2. *If f is not in IM_2, then for all sufficiently large Δ, there exists a real number $c > 1$ such that $\#\mathsf{Multi2Spin}_{\Delta,c}(f)$ is NP-hard.*

Theorems 17 and 18 together achieve the desired classification of $\#\mathsf{Multi2Spin}_\Delta(f)$ when $f \in IM_2$ as well as the strong inapproximability results when $f \notin IM_2$. Before presenting a more elaborate sketch of the proof of Theorem 17, it will be instructive to give the main ideas behind both proofs.

To prove Theorem 17, our proof departs from the previous approaches in the related works [10] and [11]. In these works, f was used to directly encode a binary function which was feasible because of the presence of equality in [10] and the symmetry of f in [11]. Instead, we take a much more painstaking combinatorial approach by using induction on the arity of the function f.

The base case of the induction (proving Theorem 17 for arity-2 functions) is fairly simple to handle, so let us focus on the induction step. The rough idea, to put the induction hypothesis to work, is to study whether f supports pinning-to-0 or pinning-to-1; then, provided that at least one these pinnings is available, we need to pin appropriately some arguments of f to obtain a function h of smaller arity. Our goal is then to ensure that h is non-affine; then, we can invoke the induction hypothesis and obtain that h either supports perfect equality or simulates a hard function. From there, since h was obtained by pinning some arguments of f, we will obtain by a transitivity argument (cf. Lemma 33 in the full version) that f either supports perfect equality or f simulates the same hard function as h. (Note, in the case where h supports perfect equality, to conclude that f supports perfect equality, we need to ensure that the pinnings of f used to obtain h were perfect.)

Determining which arguments of f need to be pinned is the most challenging aspect of this scheme. Our method for reducing the number of functions under consideration is to symmetrise f in a natural way and obtain a new function f^* which is now symmetric (see definitions in § 6). Then, it turns out that there are seven possibilities for the function f^* which we need to consider in detail. That is, when the symmetrisation of f is one of these seven functions, we have to figure out whether f supports perfect equality and, if not, work out the combinatorial structure of f and pinpoint which arguments are suitable to be pinned. The details of the argument can be found in the full version of this paper.

The proof of Theorem 18, where f supports perfect equality, basically follows the approach of [10]. However, to get the stronger inapproximability results, we have to take a detour studying self-dual functions (functions whose value does not change when we complement their arguments). We show that if f is self-dual then it simulates a hard function (Theorem 46 of the full version). The problem with self-dual functions is that they do not support pinning-to-0 or pinning-to-1, so we are not able to use the relevant results from [10]. After proving Theorem 46 and demonstrating (Lemma 42) that "implementations in CSPs" work in the repeat-free setting when f supports perfect equality, the techniques of [10] can be adapted to get Theorem 18.

6 A partial sketch of the proof of Theorem 17

Let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function. For $S \subseteq [k]$, χ_S denotes the characteristic vector of S, which is the length-k Boolean vector such that, for all $i \in [k]$, the i-th bit of χ_S is 1 iff $i \in S$. $\Omega_f = \{S \subseteq [k] \mid \chi_S \in R_f\}$. The function f is said to be *semi-trivial* iff there is a set S such that $\Omega_f = \{T \mid S \subseteq T \subseteq [k]\}$ or $\Omega_f = \{T \mid T \subseteq S\}$. Every semi-trivial Boolean function is affine. Let P_k denote the set of all permutations $\pi : [k] \to [k]$. Then the symmetrisation f^* of f is the function $f^* : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ defined by $f^*(x_1, \ldots, x_k) = \prod_{\pi \in P_k} f(x_{\pi(1)}, \ldots, x_{\pi(k)})$. For $s \in \{0,1\}$, let $\delta_s : \{0,1\} \to \{0,1\}$ be the Boolean function defined by $\delta_s(s) = 1$ and $\delta_s(1 \oplus s) = 0$. Define $f_{i \to s}$ to be the function obtained from f by pinning the i-th argument of f to s, i.e. $f_{i \to s}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k) = \sum_{x_i \in \{0,1\}} f(x_1, \ldots, x_k) \cdot \delta_s(x_i)$. Similarly, for $S, T \subseteq [k]$, let $f_{S \to 0, T \to 1}$ be the $(k - |S \cup T|)$-ary function obtained from f by pinning the arguments in S to 0 and the arguments in T to 1. So if $\mathbf{x}'$ denotes the $|S \cup T|$-ary vector containing all x_i with $i \in S \cup T$ and $\mathbf{x}''$ denotes the $k - |S \cup T|$-ary vector containing all x_i with $i \in [k] \setminus S \cup T$, $f_{S \to 0, T \to 1}(\mathbf{x}'') = \sum_{\mathbf{x}' \in \{0,1\}^{|S \cup T|}} f(x_1, \ldots, x_k) \cdot \prod_{i \in S} \delta_0(x_i) \cdot \prod_{j \in T} \delta_1(x_j)$. If $S = \emptyset$ or $T = \emptyset$, we will omit $S \to 0$ or $T \to 1$ from the notation.

Partial Proof Sketch. We prove the theorem by induction on the arity of f. The base case, $k = 2$, is covered in the full version. For the induction step, assume $k \geq 3$. The inductive hypothesis is that for all $2 \leq k' < k$, all k'-ary functions f' satisfy at least one of the three propositions in the statement of the theorem. We now prove that an arbitrary $f : \{0,1\}^k \to \{0,1\}$ also satisfies at least one of the propositions. The easy case is when f^* is not affine. In this case, the work of [11] shows that f^* either simulates a hard function (in which case f simulates the hard function as well) or f^* supports perfect equality (in which case f does as well). The bulk of the proof deals with the case where f^* is affine. It turns out that there are seven possible symmetric affine functions f^*. To illustrate the ideas, we consider just one of them here. So from now on (to cover this special case), suppose that, for all $\mathbf{x} \in \{0,1\}^k$, $f^*(\mathbf{x}) = 0$.

We prove in the full version that either f supports perfect equality or f supports both perfect pinning-to-0 and perfect pinning-to-1. If f supports perfect equality, then we are done, so suppose from now on that f supports both perfect pinning-to-0 and perfect pinning-to-1.

We will show below that at least one of the following items holds. (1) f is affine (so we are finished), (2) there exist $S, T \subseteq [k]$ such that $f_{S \to 0, T \to 1}$ is not affine, (3) f supports perfect equality (so we are finished), or (4) f simulates a hard function (so we are finished). In situation (2), since $f_{S \to 0, T \to 1}$ is not affine, it must support perfect equality or simulate a hard function g by the induction hypothesis. So we finish by showing (Lemma 33) that f either supports perfect equality or simulates the same hard function g. We now discuss how to show that at least one of the four items holds.

We prove in the full version (Lemma 39) that for all $W \in \Omega_f$, $f_{\overline{W} \to 0}$ is semi-trivial (otherwise one of the items holds). Choose $S \in \Omega_f$ such that $|S|$ is as large as possible. Let $h = f_{\overline{S} \to 0}$. Since h is semi-trivial (by taking $W = S$ above), we claim that there is a T satisfying $\emptyset \subset T \subseteq S$ such that $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$. (To see this, note that the definition of semi-trivial implies that there is a subset T of S such that either $\Omega_h = \{U \mid U \subseteq T\}$ or $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$. The former is impossible since $\emptyset \notin \Omega_h$ since $h(\mathbf{0}) = f(\mathbf{0}) = f^*(\mathbf{0}) = 0$. Also, in the latter case, T is not empty because, once again, $\emptyset \notin \Omega_h$.)

Case 1. *Suppose that* $\forall X \in \Omega_f$, $T \subseteq X$: Recall that T is non-empty. Also, for every $i \in T$, $\{i\} \cup \Omega_{f_{i \to 1}} = \Omega_f$ so either f is affine (item (1)) or $f_{i \to 1}$ is not affine (item (2)).

Now, if Case 1 does not hold then there is an $X \in \Omega_f$ such that $T \setminus X$ is non-empty. Since $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$ we conclude that $X \notin \Omega_h$. Since $h = f_{\overline{S} \to 0}$ we conclude that $X \setminus S$ is non-empty. Thus, the only other case to consider is as follows.

Case 2. *Suppose that there is an $X \in \Omega_f$ such that $T \setminus X$ and $X \setminus S$ are both non-empty:* Let $\Psi = \{X \in \Omega_f \mid T \setminus X \neq \emptyset \text{ and } X \setminus S \neq \emptyset\}$, $a = \min\{|T \setminus X| : X \in \Psi\}$, and $b = \min\{|X \setminus S| : X \in \Psi \text{ and } |T \setminus X| = a\}$. Choose $R \in \Psi$ with $|T \setminus R| = a$ and $|R \setminus S| = b$. Now before proceeding, we use the sets S, T and R to partition $[k]$. Specifically, let $A = \{i \in [k] \mid i \in S, i \in T, i \notin R\}$, $B = \{i \in [k] \mid i \in S, i \in T, i \in R\}$, $C = \{i \in [k] \mid i \in S, i \notin T, i \notin R\}$, $D = \{i \in [k] \mid i \in S, i \notin T, i \in R\}$, $E = \{i \in [k] \mid i \notin S, i \notin T, i \notin R\}$ and $F = \{i \in [k] \mid i \notin S, i \notin T, i \in R\}$. It is clear from the definitions that the sets A, B, C, D, E and F are disjoint. Also, since $T \subseteq S$, they partition $[k]$. From the definitions, $A = T \setminus R$ and $F = R \setminus S$ so, by the choice of R, A and F are non-empty. Let $g = f_{C \cup E \to 0, B \cup D \to 1}$.

By definition, every element of Ω_g is a subset of $A \cup F$. Also, for $Y \subseteq A \cup F$, "$Y \in \Omega_g$" means the same thing as "$Y \cup B \cup D \in \Omega_f$". We establish some facts before dividing the analysis into sub-cases.

Fact 1: $A \in \Omega_g$. We have $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$ and $T = A \cup B$ so $A \cup B \cup D \in \Omega_h$. Since $A \cup B \cup D \subseteq S$, this means $A \cup B \cup D \in \Omega_f$. Equivalently, $A \in \Omega_g$.

Fact 2: $F \in \Omega_g$. From the definition of R, $R \in \Omega_f$. Also, $R = B \cup D \cup F$ so $F \cup B \cup D \in \Omega_f$. Equivalently, $F \in \Omega_g$.

Fact 3: If $Y \in \Omega_g$ then either $Y \cap A \in \{\emptyset, A\}$ or $Y \cap F = \emptyset$ (or both). Suppose for contradiction that $\emptyset \subset Y \cap A \subset A$ and $Y \cap F$ is non-empty. Note that $R = B \cup D \cup F$. Let $R' = B \cup D \cup Y$. Note that $T \setminus R = A$ and $T \setminus R' = A \setminus Y \subset A$ so $|T \setminus R'| < |T \setminus R|$. We will show a contradiction to the choice of R by showing that $R' \in \Psi$. First, since $Y \in \Omega_g$, $R' \in \Omega_f$. Also, $T \setminus R' = A \setminus Y$ is non-empty and $R' \setminus S = Y \cap F$ is non-empty.

Fact 4: If $Y \in \Omega_g$ and $Y \cap A = \emptyset$ then $Y \in \{\emptyset, F\}$. Suppose for contradiction that $\emptyset \subset Y \subset F$. As in the proof of Fact 3, let $R' = B \cup D \cup Y$. Note that $T \setminus R = T \setminus R' = A$. Also, $R \setminus S = F$ and $R' \setminus S = Y$ so $|R \setminus S| > |R' \setminus S|$. Once again, we will show a contradiction to the choice of R by showing that $R' \in \Psi$. As in the proof of Fact 3, since $Y \in \Omega_g$, $R' \in \Omega_f$. Also, $T \setminus R'$ is non-empty since $T \setminus R$ is. Finally, $R' \setminus S = Y$, which is non-empty.

Fact 5: If $Y \in \Omega_g$ and $Y \cap F = \emptyset$ then $Y = A$. Since $Y \in \Omega_g$, we have $Y \cup B \cup D \in \Omega_f$. But since $Y \subseteq A$, we have $Y \cup B \cup D \subseteq S$, so $Y \cup B \cup D \in \Omega_h$. Since $\Omega_h = \{U \mid T \subseteq U \subseteq S\}$ we have $T \subseteq Y \cup B \cup D$ so $A \subseteq Y$.

Given Facts 1–5, we have only the following sub-cases.

Case 2a: $\Omega_g = \{A, F\}$. In this case, we will show that f supports perfect equality. Let H_0 be a k-tuple hypergraph, with a vertex u_0 such that $\mu_{f;H_0}(\sigma_{u_0} = 0) = 1$. Let H_1 be a k-tuple hypergraph, with a vertex u_1 such that $\mu_{f;H_1}(\sigma_{u_1} = 0) = 1$. We have already noted that A is non-empty. Suppose, without loss of generality, that $1 \in A$ (otherwise, we simply re-order the arguments of $[k]$). Now let H' be the k-tuple hypergraph with vertices $v_0, v_1, \ldots, v_k$ and hyperarcs $(v_0, v_2, \ldots, v_k)$ and $(v_1, v_2, \ldots, v_k)$. Construct H from H' by doing the following:

- For every $i \in C \cup E$, take a new copy of H_0 and identify vertex u_0 with v_i.
- For every $i \in B \cup D$, take a new copy of H_1 and identify vertex u_1 with v_i.

Now since $\Omega_g = \{A, F\}$, $\mu_{f;H}(\sigma(v_0) = \sigma(v_1) = 0) = \mu_{f;H}(\sigma(v_0) = \sigma(v_1) = 1) = 1/2$. Thus, f supports perfect equality, so item (3) holds.

Case 2b: $\exists Y \in \Omega_g$ such that $Y \cap A = A$ and $Y \cap F$ is non-empty. We show in the full version that $f_{t \to 1}$ is not affine for some $t \in A$ (so item (2) holds). ◀

References

1 Ivona Bezáková, Andreas Galanis, Leslie A. Goldberg, Heng Guo, and Daniel Štefankovič. Approximation via Correlation Decay when Strong Spatial Mixing Fails. *ArXiv e-prints*, October 2015. arXiv:1510.09193.

2 Andrei A. Bulatov, Venkatesan Guruswami, Andrei Krokhin, and Dániel Marx. The Constraint Satisfaction Problem: Complexity and Approximability (Dagstuhl Seminar 15301). *Dagstuhl Reports*, 5(7):22–41, 2016. doi:10.4230/DagRep.5.7.22.

3 Jin-Yi Cai. Complexity dichotomy for counting problems. In *Language and Automata Theory and Applications – 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, pages 1–11, 2013. doi:10.1007/978-3-642-37064-9_1.

4 Jin-Yi Cai, Andreas Galanis, Leslie A. Goldberg, Heng Guo, Mark Jerrum, Daniel Štefankovič, and Eric Vigoda. #BIS-hardness for 2-spin systems on bipartite bounded degree graphs in the tree non-uniqueness region. *Journal of Computer and System Sciences*, 82(5):690–711, 2016. doi:10.1016/j.jcss.2015.11.009.

5 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted boolean #csp. *J. Comput. Syst. Sci.*, 80(1):217–236, 2014. doi:10.1016/j.jcss.2013.07.003.

6 Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996. doi:10.1006/inco.1996.0016.

7 Víctor Dalmau and Daniel K. Ford. Generalized satisfability with limited occurrences per variable: A study through delta-matroid parity. In *Mathematical Foundations of Computer Science 2003, 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003, Proceedings*, pages 358–367, 2003. doi:10.1007/978-3-540-45138-9_30.

8 Martin Dyer, Leslie A. Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003. doi:10.1007/s00453-003-1073-y.

9 Martin E. Dyer, Leslie A. Goldberg, Markus Jalsenius, and David Richerby. The complexity of approximating bounded-degree boolean #CSP. *Inf. Comput.*, 220:1–14, 2012. doi:10.1016/j.ic.2011.12.007.

10 Martin E. Dyer, Leslie A. Goldberg, and Mark Jerrum. An approximation trichotomy for boolean #csp. *J. Comput. Syst. Sci.*, 76(3-4):267–277, 2010. doi:10.1016/j.jcss.2009.08.003.

11 Andreas Galanis and Leslie A. Goldberg. The complexity of approximately counting in 2-spin systems on k-uniform bounded-degree hypergraphs. *Information and Computation*, 251:36–66, 2016.

12 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*. Oxford lecture series in mathematics and its applications. Oxford University Press, Oxford, New York, 2004. URL: http://opac.inria.fr/record=b1121618.

13 Jonathan Hermon, Allan Sly, and Yumeng Zhang. Rapid mixing of hypergraph independent set. *CoRR*, abs/1610.07999, 2016. URL: http://arxiv.org/abs/1610.07999.

14 Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000. doi:10.1006/jcss.2000.1713.

15 Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992. URL: http://www.aaai.org/ojs/index.php/aimagazine/article/view/976.

16 Jingcheng Liu and Pinyan Lu. FPTAS for counting monotone CNF. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1531–1548, 2015. doi:10.1137/1.9781611973730.101.

17 Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974. doi:10.1016/0020-0255(74)90008-5.

18 Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

19 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226, 1978. doi:10.1145/800133.804350.

20 Renjie Song, Yitong Yin, and Jinman Zhao. Counting hypergraph matchings up to uniqueness threshold. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, pages 46:1–46:29, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.46.

21 Dominic J. A. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, New York, NY, USA, 1993.

Inapproximability of the Independent Set Polynomial Below the Shearer Threshold*†‡

Andreas Galanis[1], Leslie Ann Goldberg[2], and Daniel Štefankovič[3]

1 **University of Oxford, Oxford, UK**
andreas.galanis@cs.ox.ac.uk
2 **University of Oxford, Oxford, UK**
leslie.goldberg@cs.ox.ac.uk
3 **University of Rochester, Rochester, NY, USA**
stefanko@cs.rochester.edu

Abstract

We study the problem of approximately evaluating the independent set polynomial of bounded-degree graphs at a point λ. Equivalently, this problem can be reformulated as the problem of approximating the partition function of the hard-core model with activity λ on graphs G of maximum degree Δ. For $\lambda > 0$, breakthrough results of Weitz and Sly established a computational transition from easy to hard at $\lambda_c(\Delta) = (\Delta-1)^{(\Delta-1)}/(\Delta-2)^{\Delta}$, which coincides with the tree uniqueness phase transition from statistical physics.

For $\lambda < 0$, the evaluation of the independent set polynomial is connected to the problem of checking the conditions of the Lovász Local lemma (LLL) and applying its algorithmic consequences. Shearer described the optimal conditions for the LLL and identified the threshold $\lambda^*(\Delta) = (\Delta-1)^{\Delta-1}/\Delta^{\Delta}$ as the maximum value p such that every family of events with failure probability at most p and whose dependency graph has maximum degree Δ has nonempty intersection. Very recently, Patel and Regts, and Harvey et al. have independently designed FPTASes for approximately computing the partition function whenever $|\lambda| < \lambda^*(\Delta)$.

Our main result establishes for the first time a computational transition at the Shearer threshold. Namely, we show that for all $\Delta \geq 3$, for all $\lambda < -\lambda^*(\Delta)$, it is NP-hard to approximate the partition function on graphs of maximum degree Δ, even within an exponential factor. Thus, our result, combined with the algorithmic results for $\lambda > -\lambda^*(\Delta)$, establishes a phase transition for negative activities. In fact, we now have a complete picture for the complexity of approximating the partition function for all $\lambda \in \mathbb{R}$ and all $\Delta \geq 3$, apart from the critical values.

1. For $-\lambda^*(\Delta) < \lambda < \lambda_c(\Delta)$, there exists an FPTAS for approximating the partition function with activity λ on graphs G of maximum degree Δ.
2. For $\lambda < -\lambda^*(\Delta)$ or $\lambda > \lambda_c(\Delta)$, it is NP-hard to approximate the partition function with activity λ on graphs G of maximum degree Δ, even within an exponential factor.

Rather than the tree uniqueness threshold of the positive case, the phase transition for negative activities corresponds to the existence of zeros for the partition function of the tree below $-\lambda^*(\Delta)$.

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

Keywords and phrases approximate counting, independent set polynomial, Shearer threshold

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.28

* The full version of the paper is available at arxiv.org/abs/1612.05832. The theorem numbering here matches the full version.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

‡ Research supported by NSF grant CCF-0910415.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 28; pp. 28:1–28:13

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

1 Introduction

The independent set polynomial is a fundamental object in computer science which has been studied with various motivations. From an algorithmic viewpoint, the evaluation of this polynomial is crucial for determining the applicability of the Lovász Local Lemma and thus obtaining efficient algorithms for both finding [14] and approximately counting [6, 12] combinatorial objects with specific properties.

The independent set polynomial also arises in statistical physics, where it is called the hard-core partition function. Given a graph G, the value of the independent set polynomial of G at a point λ is equal to the value of the partition function of the hard-core model where the so-called "activity parameter" is equal to λ. We use the following notation. Given a graph G, let $\mathcal{I}_G$ denote the set of independent sets in G. The weight of an independent set $I \in \mathcal{I}_G$ is given by $\lambda^{|I|}$. The hard-core partition function with parameter λ is defined as $Z_G(\lambda) := \sum_{I \in \mathcal{I}_G} \lambda^{|I|}$.

The hard-core model has attracted significant interest in computer science during recent years, due to the pioneering results by Weitz and Sly which established that the computational complexity of approximating the partition function undergoes a transition that coincides with the uniqueness phase transition in statistical physics. Namely, for $\Delta \geq 3$, let $\lambda_c(\Delta) := (\Delta-1)^{\Delta-1}/(\Delta-2)^{\Delta}$. Weitz [22] designed an FPTAS for approximating the partition function on graphs G of maximum degree Δ when the activity parameter λ is in the range $0 < \lambda < \lambda_c(\Delta)$. On the other hand, Sly [19] showed that approximating the partition function for $\lambda > \lambda_c(\Delta)$ is NP-hard (see [20] for the refinement stated here). The threshold $\lambda_c(\Delta)$ coincides with the uniqueness threshold of the infinite Δ-regular tree and it captures whether root-to-leaf correlations persist, or decay exponentially, as the height of the tree goes to infinity. This beautiful connection between computational complexity and phase transitions has lead to a classification of the complexity of approximating the partition function of general antiferromagnetic 2-spin systems on graphs of maximum degree Δ (see [11, 18] for the algorithmic side and [20, 3] for the hardness side).

Our goal in this paper is to determine whether a computational transition takes place for negative activities as well, i.e., when $\lambda < 0$. Interestingly, the evaluation of the independent set polynomial for $\lambda < 0$ has significant algorithmic interest due to its connection with the Lovász Local Lemma (LLL) and, more precisely, to the problem of checking when the LLL applies. We will review this well-known connection shortly; prior to that, we introduce the Shearer threshold, which is relevant for our work.

Shearer, as part of his work [17] on the LLL, implicitly established that for every $\Delta \geq 2$, there is a threshold $\lambda^*(\Delta)$, given by $\lambda^*(\Delta) = (\Delta-1)^{\Delta-1}/\Delta^{\Delta}$, such that

1. for all $\lambda \geq -\lambda^*(\Delta)$, for all graphs G of maximum degree Δ, it holds that $Z_G(\lambda) > 0$.
2. for all $\lambda < -\lambda^*(\Delta)$, there exists a graph G of maximum degree Δ such that $Z_G(\lambda) \leq 0$.

We refer to the point $-\lambda^*(\Delta)$ as the *Shearer threshold*. Similarly to the positive case, the Δ-regular tree plays a role in determining the location of the Shearer threshold, in the sense that for all $\lambda < -\lambda^*(\Delta)$, the truncation of the tree at an appropriate height yields a (finite) tree T of maximum degree Δ such that $Z_T(\lambda) \leq 0$. Scott and Sokal [16] were the first to realise the relevance of Shearer's work to the phase transitions of the hard-core model, and to make explicit Shearer's contribution in this context. They further developed these ideas to study the analyticity of the logarithm of the partition function in the complex plane.

From an algorithmic viewpoint, the Shearer threshold is tacitly present in most, if not all, applications of the (symmetric) LLL. In particular, Shearer [17] proved that $\lambda^*(\Delta)$ is the maximum value p such that every family of events, with failure probability at most p and

with a dependency graph of maximum degree Δ, has nonempty intersection. This simple characterisation is a corollary of far more elaborate conditions formulated in the same work that determine whether a dependency graph falls into the scope of the LLL. To date, no polynomial-time algorithm has been presented that, given as input a dependency graph G of maximum degree Δ, decides whether Shearer's conditions are satisfied when the failure probabilities of some events exceed the threshold $\lambda^*(\Delta)$ and it is very plausible that none exists (see for example [8, Section 4] for results in this direction).

Very recently, there have been two independent works that study the Shearer threshold from an approximate counting perspective. In particular, Patel and Regts [15] and Harvey, Srivastava, and Vondrák [8] (see also [21]) designed FPTASes, using different techniques, that approximate $Z_G(\lambda)$ on graphs G of maximum degree Δ when $-\lambda^*(\Delta) < \lambda < 0$ (and also for complex values λ with $|\lambda| < \lambda^*(\Delta)$). Thus, not only is it trivial to decide whether $Z_G(\lambda)$ is positive above the Shearer threshold, but also it is computationally easy to approximate $Z_G(\lambda)$ within an arbitrarily small polynomial relative error (see [8] for extensions to the multivariate partition function). Apart from partial results in [8] which we shall review shortly, these works left open the regime $\lambda < -\lambda^*(\Delta)$. In light of their results, it is natural to ask whether the Shearer threshold has a computational complexity significance for the problem of approximating $Z_G(\lambda)$ when $\lambda < 0$, analogous to the role that the tree uniqueness threshold has for $\lambda > 0$.

In this work, we answer this question by showing that, for all $\Delta \geq 3$, for all $\lambda < -\lambda^*(\Delta)$, it is NP-hard to approximate $|Z_G(\lambda)|$, even within an exponential factor. To formally state our result, we define the following problem which has three parameters–the activity λ, a degree bound Δ, and a value $c > 1$ which specifies the desired accuracy of the approximation.

Name $\#\mathsf{HardCore}(\lambda, \Delta, c)$.
Instance An n-vertex graph G with maximum degree at most Δ.
Output A number $\widehat{Z}$ such that $c^{-n}|Z_G(\lambda)| \leq \left|\widehat{Z}\right| \leq c^n|Z_G(\lambda)|$.

We now formally state our result.

▶ **Theorem 1.** *Let $\Delta \geq 3$ and $\lambda < -\lambda^*(\Delta)$. Then there exists a constant $c > 1$ such that $\#\mathsf{HardCore}(\lambda, \Delta, c)$ is NP-hard, i.e., it is NP-hard to approximate $|Z_G(\lambda)|$ on graphs G of maximum degree at most Δ, even within an exponential factor.*

The previous known result for the inapproximability of the partition function for $\lambda < 0$ was given in [8, Theorem 4.4] which applies for $\Delta \geq 62$ and $\lambda < -39/\Delta$. Theorem 1 therefore vastly tightens that result, by showing a strong inapproximability result all the way to the Shearer threshold for all degree bounds $\Delta \geq 3$.

To elucidate the content of Theorem 1, we remark that, combined with the algorithmic results of [8, 15], it establishes for the first time a sharp computational transition at the Shearer threshold for negative activities. In fact, we now have a complete picture for the complexity of approximating $Z_G(\lambda)$ for all $\lambda \in \mathbb{R}$ apart from the critical values $-\lambda^*(\Delta)$ and $\lambda_c(\Delta)$.

1. For $-\lambda^*(\Delta) < \lambda < \lambda_c(\Delta)$, there exists an FPTAS for approximating $Z_G(\lambda)$ on graphs G of maximum degree Δ; this follows by [8, 15] for $-\lambda^*(\Delta) < \lambda < 0$ and by [22] for $0 < \lambda < \lambda_c(\Delta)$. (The case $\lambda = 0$ is trivial since $Z_G(\lambda) = 1$ for all graphs G.)
2. For $\lambda < -\lambda^*(\Delta)$ or $\lambda > \lambda_c(\Delta)$, it is NP-hard to approximate $|Z_G(\lambda)|$ on graphs G of maximum degree Δ, even within an exponential factor; this follows by Theorem 1 for $\lambda < -\lambda^*(\Delta)$ and by [20] for $\lambda > \lambda_c(\Delta)$.

While both of the thresholds $-\lambda^*(\Delta)$ and $\lambda_c(\Delta)$ come from the infinite Δ-regular tree, they are of different nature: the Shearer threshold marks the point where the partition function of the tree of appropriate height eventually becomes negative, while the uniqueness threshold

marks the point where correlations between the root and the leaves persist, as the height of the tree grows.

The interplay between the zeros of graph polynomials and the complexity of approximating partition functions has appeared before in the approximate counting literature, see for example [4, 5]. However, one of the main differences in our present setting is the constant degree bound Δ, which significantly restricts the power of "thickening". One might then think that perhaps Sly's technique for establishing inapproximability above the uniqueness threshold might be relevant; this would entail analysing the partition function of random bipartite Δ-regular graphs for negative activities using moment analysis, which, to say the least, quickly runs into severe problems. Working around these difficulties for degrees as low as $\Delta = 3$ is one the technical contributions of our work–see Section 1.1 for a high-level outline.

We conclude this introductory section by outlining very briefly the series of works that have established the Shearer threshold as an algorithmic benchmark. Beck [2] gave the first algorithmic application of the LLL, albeit with significantly worse guarantees than the non-constructive version; three decades later, Moser [13] and Moser and Tardos [14] succeeded in giving elegant, constructive analogues of the vanilla LLL; Shearer's conditions were finally used in full generality to give a constructive proof of the LLL by Kolipaka and Szegedy [10], which yielded as a corollary efficient algorithms up to the Shearer threshold. See also [1, 7, 9] for recent algorithmic extensions of the LLL (and a more thorough overview of the LLL literature) and see [6, 12] for new applications of the LLL in approximate counting.

1.1 Proof outline and organisation

At a very high level, to prove Theorem 1 for an activity $\lambda < -\lambda^*(\Delta)$, our strategy is to transform λ into a "nicer" activity. Our key technical lemma, stated as Lemma 4 in Section 2, shows how to simulate a dense set of activities on the real line using graphs of maximum degree Δ as gadgets. As we shall explain later in detail, this lemma crucially uses the assumption that $\lambda < -\lambda^*(\Delta)$ by utilising trees of appropriate depth and combining them in suitable graph constructions that respect the degree bound Δ. Once Lemma 4 is in place, some extra care is needed to obtain the inapproximability results for $\Delta = 3$. Our approach is to construct binary gadgets and use inapproximability results for antiferromagnetic 2-spin systems on 3-regular graphs.

The paper is organised in two parts. In the first part, which is in Section 2, we state our key Lemma 4 and then show how to use it to conclude the inapproximability results of Theorem 1. In the second part, which is in Section 3, we present an overview of the proof of Lemma 4 and then give, in more detail, the proofs of some indicative lemmas.

2 Proof of Theorem 1

To give some rough intuition for our main proof technique of Theorem 1, suppose that we are given a degree bound $\Delta \geq 3$ and an activity $\lambda < -\lambda^*(\Delta)$. We will pursue the freedom to "change" the activity λ to a "nicer" activity λ' by using a suitable graph of maximum degree Δ. We will refer to this construction as *implementing* the activity λ' (cf. Definition 3 for the formal notion that is used throughout the paper). Our reduction for the proof of Theorem 1 is designed so that we need to implement just two well-chosen values of λ'. Using these two activities carefully so that we do not increase the degree Δ, we will construct binary gadgets (i.e., gadgets acting on edges) that will allow us to get our NP-hardness results by reducing from an appropriate (antiferromagnetic) 2-spin model on 3-regular graphs.

To illustrate more precisely the relevant ideas, we will need a few quick definitions. Let $\lambda \in \mathbb{R}$ and $G = (V, E)$ be an arbitrary graph. For a vertex $v \in V$, we will denote

$$Z^{\mathsf{in}}_{G,v}(\lambda) := \sum_{I \in \mathcal{I}_G;\, v \in I} \lambda^{|I|}, \quad Z^{\mathsf{out}}_{G,v}(\lambda) := \sum_{I \in \mathcal{I}_G;\, v \notin I} \lambda^{|I|}.$$

Thus, $Z^{\mathsf{in}}_{G,v}(\lambda)$ is the contribution to the partition function $Z_G(\lambda)$ from those independent sets $I \in \mathcal{I}_G$ such that $v \in I$; similarly, $Z^{\mathsf{out}}_{G,v}(\lambda)$ is the contribution to $Z_G(\lambda)$ from those $I \in \mathcal{I}_G$ such that $v \notin I$. We can now formalise the notion of implementation.

▶ **Definition 2.** Let $\lambda \in \mathbb{R}_{\neq 0}$. We say that the graph G implements the activity $\lambda' \in \mathbb{R}$ with accuracy $\epsilon > 0$ if there is a vertex v in G such that $Z^{\mathsf{out}}_{G,v}(\lambda) \neq 0$ and

1. the degree of vertex v in G is 1,
2. it holds that $\left| \dfrac{Z^{\mathsf{in}}_{G,v}(\lambda)}{Z^{\mathsf{out}}_{G,v}(\lambda)} - \lambda' \right| \leq \epsilon$.

We will refer to the vertex v as the terminal of G. When Item 2 holds with $\epsilon = 0$, then we will just say that G implements the activity λ'.

▶ **Definition 3.** Let $\Delta \geq 2$ be an integer and $\lambda \in \mathbb{R}_{\neq 0}$. We say that (Δ, λ) *implements* the activity $\lambda' \in \mathbb{R}$ if there is a graph G of maximum degree at most Δ which implements the activity λ'.

More generally, we say that (Δ, λ) implements a set of activities $S \subseteq \mathbb{R}$, if for every $\lambda' \in S$ it holds that (Δ, λ) implements λ'.

Our main lemma to prove Theorem 1 is the following, whose proof is given in Section 3 (there, we also give an overview of the proof).

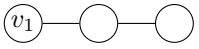
▶ **Lemma 4.** *Let $\Delta \geq 3$ and $\lambda < -\lambda^*(\Delta)$. Then, for every $\lambda' \in \mathbb{R}$, for every $\epsilon > 0$, there exists a graph G of maximum degree at most Δ that implements λ' with accuracy ϵ. In other words, (Δ, λ) implements a set of activities S which is dense in $\mathbb{R}$.*

We remark here that Lemma 4 fails for $\lambda > -\lambda^*(\Delta)$. For example, for $\lambda \geq 0$, it is not hard to see that $0 \leq Z^{\mathsf{in}}_{G,v}(\lambda)/Z^{\mathsf{out}}_{G,v}(\lambda) \leq \lambda$ for all graphs G and all vertices v in G. Moreover, in the regime $\lambda > -\lambda^*(\Delta)$, Scott and Sokal [16] have shown that $Z^{\mathsf{in}}_{G,v}(\lambda)/Z^{\mathsf{out}}_{G,v}(\lambda) > -1$ for all graphs G of maximum degree Δ (and all vertices v in G). This lower bound (in various forms) was also a key ingredient in the approximation algorithms of [8, 15].

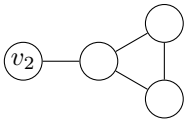
We also remark that Lemma 4 does not give any quantitative guarantees on the dependence of the size of the graph G with respect to $\lambda, \lambda', 1/\epsilon$. This is by design: such estimates will not be important for us since our reduction for Theorem 1 invokes Lemma 4 for just two constant values of λ' with some small constant $\epsilon > 0$ (the particular values depend on λ but not on the input). In particular, for our applications of Lemma 4 in the proof of Theorem 1, the sizes of the relevant graphs G will be bounded by a constant (depending on λ).

2.1 The hard-core model with non-uniform activities

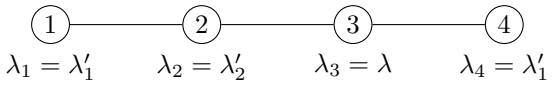
Implementing activities can be thought of as constructing unary gadgets that allow modification of the activity at a particular vertex v. We will use the implemented activities to simulate a more general version of the hard-core model with non-uniform activities. In particular, let $G = (V, E)$ be a graph and $\boldsymbol{\lambda} = \{\lambda_v\}_{v \in V}$ be a real vector; we associate to every vertex $v \in V$ the activity λ_v. The hard-core partition function with activity vector $\boldsymbol{\lambda}$ is defined as $Z_G(\boldsymbol{\lambda}) = \sum_{I \in \mathcal{I}_G} \prod_{v \in I} \lambda_v$. Note that the standard hard-core model with activity λ is obtained from this general version by setting all vertex activities equal to λ. For a vertex



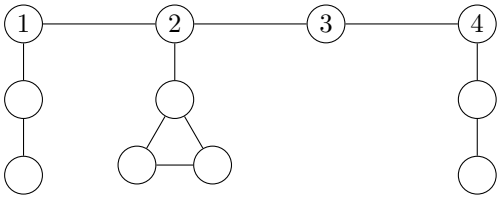
(a) The graph G_1 with terminal v_1 implementing an activity λ_1'



(b) The graph G_2 with terminal v_2 implementing an activity λ_2'



(c) The graph G with activity vector $\boldsymbol{\lambda}$



(d) The graph G' with uniform activity λ

Figure 1 An illustrative depiction of the construction in the statement of Lemma 5. The graphs G_1, G_2 in Figures 1a, 1b implement the activities λ_1', λ_2', respectively, i.e., $\frac{Z^{\mathsf{in}}_{G_1,v_1}(\lambda)}{Z^{\mathsf{out}}_{G_1,v_1}(\lambda)} = \lambda_1'$ and $\frac{Z^{\mathsf{in}}_{G_2,v_2}(\lambda)}{Z^{\mathsf{out}}_{G_2,v_2}(\lambda)} = \lambda_2'$ for some $\lambda_1', \lambda_2' \in \mathbb{R}$. In Figure 1c, we have a graph G with non-uniform activities $\{\lambda_i\}_{i\in[4]}$ such that $\lambda_i \in \{\lambda, \lambda_1', \lambda_2'\}$ for $i \in [4]$. By sticking onto G the graphs G_1, G_2 as in Figure 1d, we obtain the graph G'. Note that the vertex whose activity was equal to λ was not modified.

$v \in V$, we define $Z^{\mathsf{in}}_G(\boldsymbol{\lambda})$ and $Z^{\mathsf{out}}_G(\boldsymbol{\lambda})$ for the non-uniform model analogously to $Z^{\mathsf{in}}_G(\lambda)$ and $Z^{\mathsf{out}}_G(\lambda)$ for the uniform model, respectively.

The following lemma connects the partition function $Z_G(\boldsymbol{\lambda})$ with non-uniform activities to the hard-core partition function with uniform activity λ. Roughly, whenever all the activities in the activity vector $\boldsymbol{\lambda}$ can be implemented, we can just stick graphs on the vertices of G which implement the corresponding activities in $\boldsymbol{\lambda}$ (if a vertex activity equals λ, no action is required).

▶ **Lemma 5.** *Let $\lambda \in \mathbb{R}_{\neq 0}$, let $t \geq 1$ be an arbitrary integer, and let $\lambda_1', \ldots, \lambda_t' \in \mathbb{R}$. Suppose that, for $j \in [t]$, the graph G_j with terminal v_j implements the activity λ_j', and let $C_j := Z^{\mathsf{out}}_{G_j,v_j}(\lambda)$. Then, the following holds for every graph $G = (V, E)$ and every activity vector $\boldsymbol{\lambda} = \{\lambda_v\}_{v\in V}$ such that $\lambda_v \in \{\lambda, \lambda_1', \ldots, \lambda_t'\}$ for every $v \in V$.*

For $j \in [t]$, let $V_j := \{v \in V \mid \lambda_v = \lambda_j'\}$. Consider the graph G' obtained from G by attaching, for every $j \in [t]$ and every vertex $v \in V_j$, a copy of the graph G_j to the vertex v and identifying the terminal v_j with the vertex v (see Figure 1). Then, for $C := \prod_{j=1}^{t} C_j^{|V_j|}$, it holds that $Z_{G'}(\lambda) = C \cdot Z_G(\boldsymbol{\lambda})$.

▶ Remark. Note that, in the construction of Lemma 5, every vertex $v \in G$ with $\lambda_v = \lambda$ maintains its degree in G' (in fact, the neighbourhood of such a vertex v is the same in G and G'). The degree of every other vertex v in G gets increased by one. This observation will ensure in later applications of Lemma 5 that we do not blow up the degree.

2.2 Antiferromagnetic 2-spin systems on Δ-regular graphs

In our setting, where every vertex has degree at most Δ, an implementation consumes one of the Δ slots that a vertex has available to connect to other vertices. This is particularly problematic for the case where $\Delta = 3$. In the following we circumvent this problem by constructing suitable binary gadgets, so that we can use inapproximability results for computing the partition function of antiferromagnetic 2-spin systems on Δ-regular graphs.

Recall, an antiferromagnetic 2-spin system (without external field) is specified by two parameters $\beta, \gamma > 0$ such that $\beta\gamma < 1$. Let $\mathbf{M} = \{M_{ij}\}_{i,j\in\{0,1\}}$ be the matrix $\left[\begin{smallmatrix}\beta & 1\\ 1 & \gamma\end{smallmatrix}\right]$. For a graph $H = (V, E)$, configurations of the 2-spin system are assignments $\sigma : V \to \{0, 1\}$ and the weight of a configuration σ is given by $w_{H,\beta,\gamma}(\sigma) = \prod_{\{u,v\}\in E} M_{\sigma(u),\sigma(v)}$. The partition function of H is then given by

$$Z_{H,\beta,\gamma} = \sum_{\sigma:V\to\{0,1\}} w_{H,\beta,\gamma}(\sigma) = \sum_{\sigma:V\to\{0,1\}} \prod_{\{u,v\}\in E} M_{\sigma(u),\sigma(v)}.$$

For positive parameters β, γ and $c > 1$, we consider the following computational problem, where the input is a 3-regular graph H.

Name #2Spin(β, γ, c).
Instance An n-vertex graph H which is 3-regular.
Output A number $\hat{Z}$ such that $c^{-n} Z_{H,\beta,\gamma} \le \hat{Z} \le c^n Z_{H,\beta,\gamma}$.

The case $\beta = \gamma < 1$ corresponds to the well-known (antiferromagnetic) Ising model. As a corollary of results of Sly and Sun [20] (see also [3]), it is known that, for $0 < \beta = \gamma < 1/3$, there exists $c > 1$ such that #2Spin(β, β, c) is NP-hard, i.e., approximating the partition function $Z_{G,\beta,\beta}$ of the Ising model on 3-regular graphs H is NP-hard, even within an exponential factor. The following lemma is somewhat less known but follows easily from the results of [20].

▶ **Lemma 7.** *Let $\Delta = 3$ and β, γ be such that $0 < \beta, \gamma < 1/3$. Then, there exists $c > 1$ such that* #2Spin(β, γ, c) *is* NP*-hard.*

The following lemma will be used in the proof of Theorem 1 to specify the activities that we need to implement to utilise the inapproximability result of Lemma 7. It allows us to use the graph in Figure 2 as a binary gadget to simulate a 2-spin system with parameters β, γ.

▶ **Lemma 8.** *Let $\lambda < 0$. Then, there exist λ'_1, λ'_2 such that*

$$-1 - \frac{1}{6}|\lambda|^{1/3} < \lambda'_1 < -1, \qquad -1 - 2\lambda < \lambda'_2 < -1 - 2\lambda + \frac{\lambda'_1\lambda}{1 + \lambda'_1 + 3|\lambda|^{1/3}}. \tag{1}$$

For all λ'_1, λ'_2 satisfying (1)*, the following parameters β, γ (defined in terms of $\lambda, \lambda'_1, \lambda'_2$)*

$$\beta = \frac{(1+\lambda'_1)\Big((1+\lambda'_1)(1+\lambda'_2+2\lambda) - 2\lambda'_1\lambda\Big)}{|\lambda|^{1/3}\Big(\lambda'_1\lambda - (1+\lambda'_1)(1+\lambda'_2+2\lambda)\Big)}, \quad \gamma = \frac{|\lambda|^{1/3}(1+\lambda'_2+2\lambda)}{\lambda'_1\lambda - (1+\lambda'_1)(1+\lambda'_2+2\lambda)}. \tag{2}$$

satisfy $0 < \beta, \gamma < 1/3$.

2.3 The reduction & Proof of Theorem 1

The reduction to obtain Theorem 1 uses a binary gadget to simulate an antiferromagnetic 2-spin system on 3-regular graphs, i.e., we will replace every edge of a 3-regular graph H with a suitable graph B which has two special vertices to encode the edge. The gadget B is given in Figure 2, the two special vertices are v_1, v_2. Note that the gadget B has nonuniform activities but this will be compensated for later by invoking Lemma 5. We thus obtain the following lemma (whose proof is in the full version).

▶ **Lemma 9.** *Let $\lambda < 0$ and $\lambda'_1, \lambda'_2 \in \mathbb{R}$ satisfy* (1)*. Then, for β, γ as in* (2)*, the following holds. For every 3-regular graph $H = (V_H, E_H)$ we can construct in linear time a graph $G = (V_G, E_G)$ of maximum degree 3 and specify an activity vector $\boldsymbol{\lambda} = \{\lambda_v\}_{v\in V}$ on G such that*

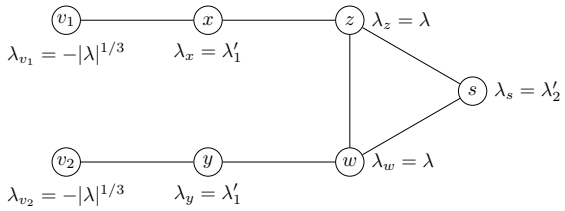


Figure 2 The binary gadget $B = (U, F)$ used in Lemma 9 to simulate an antiferromagnetic 2-spin system on 3-regular graphs. The gadget B is used to encode the edges of a 3-regular graph H. In particular, every edge $e = \{h_1, h_2\}$ of H gets replaced by a distinct copy of B, with the vertices v_1, v_2 of B getting identified with the vertices h_1, h_2 of H, respectively.

1. $Z_{H,\beta,\gamma} = Z_G(\boldsymbol{\lambda})/C^{|E_H|}$, *where* $C := |\lambda|^{1/3}\big(\lambda_1'\lambda - (1+\lambda_1')(1+\lambda_2'+2\lambda)\big) > 0$.
2. *For every vertex* v *of* G, *it holds that* $\lambda_v \in \{\lambda, \lambda_1', \lambda_2'\}$. *Moreover, if* $\lambda_v \neq \lambda$, *then* v *has degree two in* G.

Now, we are ready to prove Theorem 1.

Proof of Theorem 1 (Sketch). By Lemma 4, there are graphs G_1, G_2 of max degree Δ with terminals v_1, v_2 which implement activities λ_1', λ_2' satisfying the condition (1) of Lemma 8. For later use, set $C_1 := Z^{\mathsf{out}}_{G_1,v_1}(\lambda)$, $C_2 := Z^{\mathsf{out}}_{G_2,v_2}(\lambda)$ and note that C_1, C_2 are explicitly computable constants. Let β, γ be the parameters given by (2). By Lemma 8, it holds that $0 < \beta, \gamma < 1/3$. Thus, by Lemma 7, there exists $c > 1$ such that $\#\mathsf{2Spin}(\beta, \gamma, c)$ is NP-hard. We will use Lemmas 5 and 9 to reduce $\#\mathsf{2Spin}(\beta, \gamma, c)$ to $\#\mathsf{HardCore}(\lambda, \Delta, c')$ for some constant $c' > 1$.

Let H be a 3-regular graph which is an input graph to the problem $\#\mathsf{2Spin}(\beta, \gamma, c)$. By Lemma 9, we can construct in linear time a graph G of maximum degree 3 and specify an activity vector $\boldsymbol{\lambda} = \{\lambda_v\}_{v \in V}$ on G such that

1. $Z_{H,\beta,\gamma} = Z_G(\boldsymbol{\lambda})/C^{|E_H|}$, where $C := |\lambda|^{1/3}\big(\lambda_1'\lambda - (1+\lambda_1')(1+\lambda_2'+2\lambda)\big) > 0$.
2. For every vertex v of G, $\lambda_v \in \{\lambda, \lambda_1', \lambda_2'\}$. Also, if $\lambda_v \neq \lambda$, then v has degree two in G.

Using the graphs G_1, G_2 that implement λ_1', λ_2' respectively, we obtain from Lemma 5 that we can construct in linear time a graph $G' = (V_{G'}, E_{G'})$ of maximum degree at most Δ such that $Z_{G'}(\lambda) = C_1^{n_1} C_2^{n_2} \cdot Z_G(\boldsymbol{\lambda})$, where n_1, n_2 are the number of vertices in G whose activity equals λ_1', λ_2', respectively. Note, the fact that the maximum degree of G' is at most Δ follows from the construction of Lemma 5 and Item 2 (cf. the Remark after Lemma 5).

It follows that $Z_{H,\beta,\gamma} = Z_{G'}(\lambda)/\big(C^{|E_H|} C_1^{n_1} C_2^{n_2}\big)$. Since the size of G' exceeds the size of H only by a constant factor, there is a constant $c' > 1$ (depending only on λ) such that an approximation to $|Z_{G'}(\lambda)|$ within a factor $(c')^{|V_{G'}|}$ yields an estimate to $|Z_{H,\beta,\gamma}| = Z_{H,\beta,\gamma}$ within a factor $c^{|V_H|}$. It follows that $\#\mathsf{HardCore}(\lambda, \Delta, c')$ is NP-hard. ◀

3 Proof of Lemma 4

In this final section, we give a proof overview of Lemma 4, which is the last missing ingredient used in the proof of Theorem 1. Let us fix a degree bound $\Delta \geq 3$. Our goal is to show that for any fixed $\lambda < -\lambda^*(\Delta)$, we can implement a dense set of activities using graphs of maximum degree Δ. At a very rough level, the proof of Lemma 4 splits into two regimes:

1. when $\lambda < -\lambda^*(2) = -1/4$,
2. when $-1/4 \leq \lambda < -\lambda^*(\Delta)$.

Roughly, in regime 1, we will be able to use *paths* to implement a dense set of activities. In regime 2, we will first use a $(\Delta - 1)$-ary tree to implement an activity $\lambda' < -1/4$. Then,

using the activity λ', we will be able to use the path construction of the first regime to implement a dense set of activities.

Unfortunately, the actual proof is more intricate, since as it turns out there is a set $\mathcal{B} \subset \mathbb{R}$, dense in $(-\infty, -1/4)$, such that, if $\lambda \in \mathcal{B}$, paths exhibit a periodic behaviour in terms of implementing activities (and thus can only be used to implement a finite set of activities). The following lemma will be important in specifying the set $\mathcal{B}$ and understanding this periodic behaviour. The proof is a manipulation with trigonometric identities and can be found in Section 3.2 of the full version.

▶ **Lemma 10.** *Let $\lambda < -1/4$ and $\theta \in (0, \pi/2)$ be such that $\lambda = -1/(2\cos\theta)^2$. Then, the partition function of the path P_n with n vertices is given by*

$$Z_{P_n}(\lambda) = \frac{\sin((n+2)\theta)}{2^n(\cos\theta)^n \sin(2\theta)}.$$

The "bad" set $\mathcal{B}$ of activities (for which paths exhibit a periodic behaviour) can be read off from Lemma 10. To make this precise, let

$$\mathcal{B} := \left\{\lambda \in \mathbb{R} \mid \lambda = -\frac{1}{4(\cos\theta)^2} \text{ for some } \theta \in (0, \pi/2) \text{ which is a } \textit{rational multiple} \text{ of } \pi\right\}. \quad (3)$$

Note, for example, that $-1, -1/2, -1/3 \in \mathcal{B}$ (set $\theta = \pi/3, \pi/4, \pi/6$, respectively). For $\lambda < -1/4$, it is not hard to infer from Lemma 10 that the ratio $\frac{Z^{\mathsf{in}}_{P_n,v}(\lambda)}{Z^{\mathsf{out}}_{P_n,v}(\lambda)}$ is equal to $-\frac{1}{2\cos\theta}\frac{\sin(n\theta)}{\sin((n+1)\theta)}$. Therefore, when $\lambda \in \mathcal{B}$ or equivalently θ is a rational multiple of π, the ratio is periodic in terms of the number of vertices n in the path. On the other hand, when $\lambda < -1/4$ and $\lambda \notin \mathcal{B}$, then we can show that the ratio is dense in $\mathbb{R}$ as n varies (this follows essentially from the fact that $\{n\theta \bmod 2\pi \mid n \in \mathbb{Z}\}$ is dense on the circle when θ is irrational) and hence we can use paths to implement a dense set of activities. This is the scope of the next lemma, which is proved in Section 3.2 of the full version.

▶ **Lemma 11.** *Let $\lambda < -1/4$ be such that $\lambda \notin \mathcal{B}$. Let P_n denote a path with n vertices and let v be one of the endpoints of P_n. Then, for every $\lambda' \in \mathbb{R}$, for every $\epsilon > 0$, there exists n such that $|\frac{Z^{\mathsf{in}}_{P_n,v}(\lambda)}{Z^{\mathsf{out}}_{P_n,v}(\lambda)} - \lambda'| \le \epsilon$.*

When $\lambda \in \mathcal{B}$, we can no longer use paths to implement a dense set of activities, as we explained earlier, and we need to use a more elaborate argument. A key observation is that, for $\lambda \in \mathcal{B}$, the partition function of a path of appropriate length is equal to 0. In particular, we have the following simple corollary of Lemma 10.

▶ **Corollary 12.** *Let $\lambda < -1/4$ be such that $\lambda \in \mathcal{B}$. Denote by P_n the path with n vertices. Then, there is an integer $n \ge 1$ such that the partition function of the path P_n is zero, i.e., $Z_{P_n}(\lambda) = 0$.*

Having a path P whose partition function is 0 allows us to implement the activity -1: indeed, for an endpoint v of the path P, we have that $Z^{\mathsf{in}}_{P,v}(\lambda) + Z^{\mathsf{out}}_{P,v}(\lambda) = Z_P(\lambda) = 0$, and hence P, with terminal v, implements $\frac{Z^{\mathsf{in}}_{P,v}(\lambda)}{Z^{\mathsf{out}}_{P,v}(\lambda)} = -1$ (note, we will later ensure that P is such that $Z^{\mathsf{out}}_{P,v}(\lambda) \neq 0$). A somewhat ad-hoc gadget allows us to also implement the activity $+1$. Using these two implemented activities, -1 and $+1$, we then show how to implement *all* rational numbers using graphs whose structure resembles a caterpillar (the proof is inspired by the "ping-pong" lemma in group theory, used to establish free subgroups). We carry out this scheme in a more general setting where, instead of a path, we have a tree whose partition function is zero (this will also be relevant in the regime $\lambda > -1/4$). More precisely, we have the following lemma, whose proof is given in Section 3.1.

▶ **Lemma 13.** *Suppose that $\lambda \in \mathbb{R}_{\neq 0}$ and that T is a tree with $Z_T(\lambda) = 0$. Let d be the maximum degree of T and let $\Delta = \max\{d, 3\}$. Then, (Δ, λ) implements a dense set of activities in $\mathbb{R}$.*

We thus obtain the following throughout the regime 1 ($\lambda < -1/4$).

▶ **Lemma 14.** *Let $\lambda < -1/4$. For $\Delta = 3$, (Δ, λ) implements a dense set of activities in $\mathbb{R}$.*

Proof. We may assume that $\lambda \in \mathcal{B}$, otherwise the result follows directly from Lemma 11. For $\lambda \in \mathcal{B}$, we have by Corollary 12 a path P such that $Z_P(\lambda) = 0$. Since P has maximum degree 2, applying Lemma 13 gives the desired conclusion. ◀

Note, Lemma 14 applies only for values of λ which are far from the threshold $-\lambda^*(\Delta)$ for any $\Delta \geq 3$ and thus it should not be surprising that we can implement a dense set of activities using graphs of maximum degree 3. This highlights the next obstacle that we have to address: for general degree bounds $\Delta \geq 3$, to get all the way to the threshold $-\lambda^*(\Delta)$ we need to use graphs with maximum degree Δ (rather than just 3) to have some chance of implementing interesting activities.

Analyzing more complicated graphs for $\Delta \geq 3$ and $-1/4 \leq \lambda < -\lambda^*(\Delta)$ might sound daunting given the story for $\lambda < -1/4$, but it turns out that all we need to do is construct a graph G of maximum degree Δ that implements an activity $\lambda' < -1/4$. Then, to show that (Δ, λ) implements a dense set of activities, we only need to consider whether $\lambda' \in \mathcal{B}$. If $\lambda' \notin \mathcal{B}$, we can argue by decorating the paths from Lemma 11 using the graph G. Otherwise, if $\lambda' \in \mathcal{B}$, we can first construct a tree T of maximum degree Δ such that $Z_T(\lambda) = 0$ (by decorating the path from Lemma 12), and then invoke Lemma 13. Thus, we are left with the task of implementing an activity $\lambda' < -1/4$. For that, we combine appropriately $(\Delta - 1)$-ary trees of appropriate depth, which can be analysed relatively simply using a recursion. (A technical detail here is that, initially, we are not able to implement this boosted activity λ' in the sense of Definition 3 since the terminal of the relevant tree has degree bigger than 1; nevertheless, the degree of the terminal is at most $\Delta - 2$, so it can be combined with the paths without overshooting the degree bound Δ.) Putting together these pieces yields the following lemma (see Section 3.5 of the full version).

▶ **Lemma 15.** *Let $\Delta \geq 3$ and $-1/4 \leq \lambda < -\lambda^*(\Delta)$. Then, (Δ, λ) implements a dense set of activities in $\mathbb{R}$.*

Using Lemmas 14 and 15, the proof of Lemma 4 is immediate.

3.1 The case where the partition function of some tree is zero

In this section, we prove Lemma 13 which is an ingredient in both Lemmas 14 and 15.

We start with the following lemma, whose full proof is given in the full version. Roughly, in the proof, the implementation of -1 uses as a gadget the tree T with a leaf as the terminal; the implementation of $+1$ uses an ad-hoc gadget.

▶ **Lemma 19.** *Let $\lambda \in \mathbb{R}_{\neq 0}$ and $d \geq 2$ be a positive integer. Suppose that there exists a tree T with maximum degree d such that $Z_T(\lambda) = 0$. Then, for $\Delta = \max\{d, 3\}$, we have that (Δ, λ) implements the activities -1 and $+1$.*

The following functions f_+ and f_- will be important in what follows:

$$f_+ : \mathbb{R}\backslash\{-1\} \mapsto \mathbb{R}\backslash\{0\}, \text{ given by } f_+(x) = \frac{1}{1+x} \text{ for all } x \neq -1,$$

$$f_- : \mathbb{R}\backslash\{+1\} \mapsto \mathbb{R}\backslash\{0\}, \text{ given by } f_-(x) = \frac{1}{1-x} \text{ for all } x \neq +1.$$

▶ **Definition 20.** Let $S \subseteq \mathbb{R}$ be the set of real numbers defined as follows: $z \in S$ iff for some integer $n \geq 0$, there exists a sequence $x_0, \ldots, x_n$ such that $x_0 = 0$, $x_n = z$ and for all $i = 0, \ldots, n-1$ it holds that either $x_{i+1} = f_+(x_i)$ or $x_{i+1} = f_-(x_i)$.

The set S in Definition 20 is obtained by the following recursive procedure. Set $S_0 = \{0\}$. For $h = 0, 1, \ldots$, define S_{h+1} by first letting $S^+_{h+1} = f_+(S_h)$ and $S^-_{h+1} = f_-(S_h)$ and then setting $S_{h+1} = S^+_{h+1} \cup S^-_{h+1}$. S can then be recovered by taking the union of the sets S_h, i.e., $S = \cup^\infty_{h=0} S_h$. Our interest in the set S is due to the following lemma (proof in the full version).

▶ **Lemma 21.** *Let $\Delta \geq 3$ and $\lambda < 0$. Suppose that (Δ, λ) implements the activities -1 and $+1$. Then, (Δ, λ) also implements the set of activities $\{\lambda z \mid z \in S\}$.*

It is simple to see that all numbers in the set S of Definition 20 are rationals. Somewhat surprisingly, the following lemma asserts that S is in fact the set $\mathbb{Q}$ of all rational numbers.

▶ **Lemma 22.** *Let $S \subseteq \mathbb{R}$ be the set in Definition 20. Then, $S = \mathbb{Q}$.*

Proof. Recall that $S \subseteq \mathbb{Q}$, so we only need to argue that $\mathbb{Q} \subseteq S$. Since $0 \in S$ (by taking $n = 0$ in Definition 20) and $f_+(0) = 1$, we have that $0, 1 \in S$. Note that

$$f_-(f_-(f_+(x))) = -x \text{ for } x \neq -1, 0. \tag{4}$$

It follows that $-1 \in S$. Also, $1/2, 2 \in S$ since $f_+(f_+(0)) = 1/2$ and $f_-(f_+(f_+(0))) = 2$. Let $T := \{-1, 0, 1/2, 1, 2\}$; the arguments above established that $T \subseteq S$. Consider an arbitrary $\rho \in \mathbb{Q}$ such that $\rho \notin T$. To prove the lemma, we need to show that $\rho \in S$.

We will show that, for some integer $n \geq 0$, there is a sequence $\{\rho_i\}^n_{i=0}$ such that

(i) $\rho_0 = \rho$, $\rho_n = -1$.
(ii) $\rho_i \notin \{0, 1/2, 1\}$ for $i = 0, \ldots, n-1$.
(iii) $\rho_{i+1} = f_+(\rho_i)$ or $\rho_{i+1} = f_-(\rho_i)$ for $i = 0, \ldots, n-1$.

Before proving the existence of such a sequence, we first show how to conclude that $\rho \in S$. To do this, let $x_i := \rho_{n-i}$ for $i = 0, \ldots, n$. Properties (i)–(iii) of the sequence $\{\rho_i\}^n_{i=0}$ translate into the following properties of the sequence $\{x_i\}^n_{i=0}$:

(a) $x_0 = -1$, $x_n = \rho$.
(b) $x_i \notin \{0, 1/2, 1\}$ for $i = 1, 2, \ldots, n$.
(c) $x_i = f^{-1}_+(x_{i-1})$ or $x_i = f^{-1}_-(x_{i-1})$ for $i = 1, 2, \ldots, n$.

We show by induction on i that $x_i \in S$ for all $i = 0, \ldots, n$, which for $i = n$ gives that $\rho \in S$ (since by Item (a) we have $x_n = \rho$). For the base case $i = 0$, we have that $x_0 = -1$ by Item (a) and hence $x_0 \in S$. For the induction step, assume that $x_i \in S$ for some integer $0 \leq i \leq n-1$, our goal is to show that $x_{i+1} \in S$. The main observation is that the inverses of the functions f_- and f_+ can be obtained by composing appropriately the functions f_- and f_+. Namely, we have that

$$f^{-1}_-(x) = \frac{x-1}{x} = f_-(f_-(x)) \text{ for } x \neq 0, 1, \tag{5}$$

$$f^{-1}_+(x) = \frac{1-x}{x} = f_-(f_-(f_+(f_-(f_-(x)))))) \text{ for } x \neq 0, \tfrac{1}{2}, 1. \tag{6}$$

(5) is proved by just making the substitutions. (6) is obtained from (4) and (5), and checking when $f_-(f_-(x)) = \frac{x-1}{x}$ equals -1 and 0. Since by Items (a) and (b) we have that $x_j \neq 0, 1/2, 1$ for all $0 \leq j \leq n$ and $x_i \in S$ by the induction hypothesis, it follows by Item (c) and (5), (6) that $x_{i+1} \in S$, as wanted.

It remains to establish the existence of the sequence $\{\rho_i\}^n_{i=0}$ with the properties (i)–(iii). Consider the following set S_ρ, which is defined analogously to the set S with the only

difference that the starting point for S_ρ is the point ρ (instead of 0 that was used in the definition of S). Formally, $z \in S_\rho$ iff for some integer $n \geq 0$, there exists a sequence $\{\rho_i\}_{i=0}^n$ such that $\rho_0 = \rho$, $\rho_n = z$ and for all $i = 0, \ldots, n-1$ it holds that either $\rho_{i+1} = f_+(\rho_i)$ or $x_{i+1} = f_-(\rho_i)$. For convenience, we will call such a sequence a certificate that $z \in S_\rho$ and we will refer to n as the length of the certificate. We will show that, for any $\rho \in \mathbb{Q}$ such that $\rho \notin T = \{-1, 0, 1/2, 1, 2\}$, it holds that

$$0, 1 \notin S_\rho, \qquad -1 \in S_\rho. \tag{7}$$

By considering a certificate of smallest length that $-1 \in S_\rho$ (existence is guaranteed by (7)), we obtain a sequence $\{\rho_i\}_{i=0}^n$ that has all of the required properties (i), (ii), and (iii), see the full version for the details. Thus, in the following we focus on establishing (7). First, we show that $0, 1 \notin S_\rho$. Observe that $\rho \neq 0, 1$, so any certificate that $0, 1 \in S_\rho$ must have nonzero length. Further, the range of the functions f_+, f_- excludes 0, which implies that $0 \notin S_\rho$. Moreover, the only way that we can have $1 \in S_\rho$ is if for some $x \in S_\rho$ it holds that $f_+(x) = 1$ or $f_-(x) = 1$. Both of these mandate that $x = 0$, but $0 \notin S_\rho$ as we just showed.

The remaining bit of (7), i.e., that $-1 \in S_\rho$, will require more effort to prove. As a starting point, note that from $\rho \in \mathbb{Q}$, we have that $S_\rho \subseteq \mathbb{Q}$. Also, S_ρ is nonempty since $\rho \in S_\rho$. Thus, there exists $z^* \in S_\rho$ such that $z^* = p/q$ where p, q are integers such that $|p| + |q|$ is minimum. Since $|p| + |q|$ is minimum, it must be the case that $\gcd(p, q) = 1$.

We first prove that $z^* \in T$; note, we already know that $z^* \neq 0, 1$ since $0, 1 \notin S_\rho$ and $z^* \in S_\rho$, but keeping the values $0, 1$ into consideration will be convenient for the upcoming argument. Namely, for the sake of contradiction, assume that $z^* \notin T$, which implies in particular that $z^* \neq 0, -1$. Since $z^* \in S_\rho$, by (4), we obtain that $-z^* \in S_\rho$ as well. By switching to $-z^*$ if necessary, we may thus assume that z^* is positive and hence that $p, q > 0$, i.e., that both p, q are positive integers. Since $z^* \neq 1$ (from $z^* \notin T$), we have that $p \neq q$. For each of the cases $p > q$ and $p < q$, we obtain a contradiction to the minimality of $p + q$ by constructing $z' = p'/q' \in S_\rho$ with p', q' positive integers such that $0 < p' + q' < p + q$.

Case 1. $p > q$. Since $z^* \neq 1, 2$ (from $z^* \notin T$), we have that $p/q \neq 1$ and $f_-(p/q) = \frac{q}{q-p} \neq 0, -1$, so by (4) we have that $f_-(f_-(f_+(f_-(p/q)))) = \frac{q}{p-q}$. Thus, letting $p' = q$ and $q' = p - q$ yields $z' = p'/q' \in S_\rho$ with $p' > 0, q' > 0$ and $0 < p' + q' < p + q$.

Case 2. $p < q$. Since $z^* \neq 0, 1/2, 1$ (from $z^* \notin T$), we obtain from (6) that $f_-(f_-(f_+(f_-(f_-(p/q))))) = \frac{q-p}{p}$. Thus, letting $p' = q - p$ and $q' = p$ yields $z' = p'/q' \in S_\rho$ with $p' > 0, q' > 0$ and $0 < p' + q' < p + q$.

This concludes the proof that $z^* \in T$. In fact, we can now deduce easily that $-1 \in S_\rho$. As noted earlier, we have that $z^* \neq 0, 1$ as a consequence of $0, 1 \notin S_\rho$, so in fact $z^* \in \{-1, 1/2, 2\}$. If $z^* = -1$, then we automatically have that $-1 \in S_\rho$ since z^* was chosen to be in S_ρ. If $z^* = 2$, then we have that $2 \in S_\rho$ and hence $f_-(2) = -1 \in S_\rho$ as well. Finally, if $z^* = 1/2$, we have that $1/2 \in S_\rho$ and hence $f_-(f_-(1/2)) = -1 \in S_\rho$. Thus, it holds that $-1 \in S_\rho$, which completes the proof of (7) and hence the proof of Lemma 22. ◀

Combining Lemmas 19, 21 and 22, we obtain Lemma 13.

References

1 Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász Local Lemma. *J. ACM*, 63(3):Article No. 22, July 2016.

2 József Beck. An algorithmic approach to the Lovász Local Lemma. I. *Random Structures & Algorithms*, 2(4):343–365, 1991.

3 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *Combinatorics, Probability and Computing*, 25(4):500–559, 2016.

4 Leslie A. Goldberg and Mark Jerrum. Inapproximability of the Tutte polynomial of a planar graph. *Computational Complexity*, 21(4):605–642, 2012.

5 Leslie A. Goldberg and Mark Jerrum. The complexity of computing the sign of the Tutte polynomial. *SIAM Journal on Computing*, 43(6):1921–1952, 2014.

6 Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász Local lemma. *CoRR*, abs/1611.01647, 2016.

7 David G. Harris and Aravind Srinivasan. A constructive algorithm for the Lovász Local Lemma on permutations. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'14, pages 907–925, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

8 Nicholas J. A. Harvey, Piyush Srivastava, and Jan Vondrák. Computing the independence polynomial in Shearer's region for the LLL. *CoRR*, abs/1608.02282, 2016.

9 Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász Local Lemma via resampling oracles. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS'15, pages 1327–1346, Washington, DC, USA, 2015. IEEE Computer Society.

10 Kashyap B. R. Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC'11, pages 235–244, New York, NY, USA, 2011. ACM.

11 Liang Li, Pinyan Lu, and Yitong Yin. Correlation decay up to uniqueness in spin systems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 67–84, 2013.

12 Ankur Moitra. Approximate counting, the Lovász Local lemma and inference in graphical models. *CoRR*, abs/1610.04317, 2016.

13 Robin A. Moser. A constructive proof of the Lovász Local Lemma. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC'09, pages 343–350, New York, NY, USA, 2009.

14 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász Local lemma. *J. ACM*, 57(2):Article No. 11, 2010.

15 Viresh Patel and Guus Regts. Deterministic polynomial-time approximation algorithms for partition functions and graph polynomials. *CoRR*, abs/1607.01167, 2016.

16 Alexander D. Scott and Alan D. Sokal. The repulsive lattice gas, the independent-set polynomial, and the Lovász Local Lemma. *J. Stat. Phys.*, 118(5):1151–1261, 2005.

17 J. B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985.

18 Alistair Sinclair, Piyush Srivastava, and Marc Thurley. Approximation algorithms for two-state anti-ferromagnetic spin systems on bounded degree graphs. *J. Stat. Phys.*, 155(4):666–686, 2014.

19 Allan Sly. Computational transition at the uniqueness threshold. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS'10, pages 287–296, Washington, DC, USA, 2010. IEEE Computer Society.

20 Allan Sly and Nike Sun. Counting in two-spin models on d-regular graphs. *Ann. Probab.*, 42(6):2383–2416, 11 2014.

21 Piyush Srivastava. Approximating the hard core partition function with negative activities. Manuscript, available at `http://www.its.caltech.edu/~piyushs/`, April 2015.

22 Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC'06, pages 140–149, New York, NY, USA, 2006. ACM.

The Complexity of Holant Problems over Boolean Domain with Non-Negative Weights*†

Jiabao Lin[1] **and Hanpin Wang**[‡2]

1 **Key Laboratory of High Confidence Software Technologies (MOE), School of Electronics Engineering and Computer Science, Peking University, Beijing, China**
joblin@pku.edu.cn

2 **Key Laboratory of High Confidence Software Technologies (MOE), School of Electronics Engineering and Computer Science, Peking University, Beijing, China**
whpxhy@pku.edu.cn

Abstract

Holant problem is a general framework to study the computational complexity of counting problems. We prove a complexity dichotomy theorem for Holant problems over the Boolean domain with non-negative weights. It is the first complete Holant dichotomy where constraint functions are not necessarily symmetric.

Holant problems are indeed read-twice #CSPs. Intuitively, some #CSPs that are #P-hard become tractable when restricted to read-twice instances. To capture them, we introduce the Block-rank-one condition. It turns out that the condition leads to a clear separation. If a function set $\mathcal{F}$ satisfies the condition, then $\mathcal{F}$ is of affine type or product type. Otherwise (a) Holant($\mathcal{F}$) is #P-hard; or (b) every function in $\mathcal{F}$ is a tensor product of functions of arity at most 2; or (c) $\mathcal{F}$ is transformable to a product type by some real orthogonal matrix. Holographic transformations play an important role in both the hardness proof and the characterization of tractability.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases counting complexity, dichotomy, Holant, #CSP

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.29

1 Introduction

There has been considerable interest in several frameworks to study the complexity of counting problems. One natural framework is the counting Constraint Satisfaction Problem (#CSP) [18, 2, 19, 4, 22, 3, 8, 7, 1]. Another is Graph Homomorphism (GH) [30, 27, 21, 5, 20, 25, 6, 9], which can be seen as a special case of #CSP. Such frameworks express a large class of counting problems in the Sum-of-Product form. It is known that if P $\neq$ NP, then there exists a problem that is neither in P nor NP-complete [29]. And there is an analogue of Ladner's Theorem for the class #P. However, for these frameworks, various beautiful dichotomy theorems have been proved, classifying all problems in the broad class into those which are computable in polynomial time (in P) and those which are #P-hard. A natural question is: For how broad a class of counting problems can one prove a dichotomy theorem?

* A full version containing detailed proofs is available at https://arxiv.org/abs/1611.00975.

† This work was supported by the National Natural Science Foundation of China (Grants No. 61170299, 61370053 and 61572003).

‡ Hanpin Wang is the corresponding author.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 29; pp. 29:1–29:14

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

While GH can express many interesting graph parameters, Freedman, Lovász and Schrijver [24] showed that the number of perfect matchings of a graph cannot be represented as a homomorphism function. Inspired by holographic algorithms [32, 31], Cai, Lu and Xia [14] proposed a more refined framework called Holant Problems. Here we give a brief introduction. In this paper, constraint functions are defined over the Boolean domain, if not specified. Let $\mathcal{F}$ denote a set of algebraic complex-valued functions. A *signature grid* Ω is a tuple $(G, \mathcal{F}, \pi)$ where $G = (V, E)$ is an undirected graph, and π is a map that maps each vertex $v \in V$ to some function $f_v \in \mathcal{F}$ and its incident edges $E(v)$ to the input variables of f_v. The counting problem on Ω is to compute

$$\text{Holant}_\Omega = \sum_{\sigma: E \to \{0,1\}} \prod_{v \in V} f_v(\sigma|_{E(v)}),$$

where $\sigma|_{E(v)}$ is the restriction of σ to $E(v)$. All such signature grids constitute the set of instances of the problem Holant($\mathcal{F}$). For example, consider the problem of counting perfect matchings (#PM) on graph G. In a perfect matching, every vertex is saturated by exactly one edge. Such constraint on a vertex of degree n can be expressed as an Exact-One function $f : \{0,1\}^n \to \{0,1\}$, which takes the value 1 if and only if its input has Hamming weight 1. If every vertex is assigned such a function, then the value Holant_Ω is exactly the number of perfect matchings. Let $\mathcal{F}$ denote the set of all Exact-One functions, then Holant($\mathcal{F}$) represents the problem #PM.

The Holant framework is general enough: #CSPs can be viewed as special Holant problems where all equality functions are available [14]. However, the very generality makes it more difficult to prove a dichotomy. A function is *symmetric* if the function values only depend on the Hamming weights of inputs, like the Exact-One functions. Satisfactory progress has been made in the complexity classification of Holant problems specified by sets of symmetric functions [13, 28, 26, 11, 10]. And in the process, some unexpected tractable classes were discovered. They give many deep insights into both tractability and hardness.

It still remains open whether a complete dichotomy exists, since the definition of Holant problems does not require that constraint functions be symmetric. Such restriction is stringent and generally it is not imposed in #CSP. Cai, Lu and Xia [16] proved a dichotomy without symmetry for a special family of Holant problems, called Holant*, where all unary functions are assumed to be available. But without this assumption, as in [11], more tractable classes will be released, which makes the hardness proof very different.

We prove a dichotomy theorem for Holant problems with non-negative algebraic real weights. It is the first complete Holant dichotomy where constraint functions are not necessarily symmetric and no auxiliary function is assumed to be available. This generalizes the results on Boolean #CSP in [18, 19], and the dichotomies in [28, 11] restricted to non-negative case. Our proof starts with an infinitary condition, but finally obtains an explicit criterion (Theorem 19).

A simple observation is that, Holant problems are indeed *read-twice* #CSPs where every variable in an instance appears exactly twice (see subsection 2.4). Intuitively, some #CSPs that are #P-hard become tractable when restricted to read-twice instances. To capture them, we need insights into what makes a problem hard in #CSP. Inspired by dichotomy theorems over general domains [5, 23, 8, 7], we introduce the Block-rank-one condition for Holant problems (see subsection 7.1). It is known that non-block-rank-one structures imply hardness in #CSP. So our condition is necessary for tractability since it is imposed on the functions defined by read-twice instances. Surprisingly, on the Boolean domain, the Block-rank-one condition is also *sufficient* and leads to a clear separation:

I. **Function set $\mathcal{F}$ satisfies the condition**. Then #CSP($\mathcal{F}$) is in P, and hence its subproblem Holant($\mathcal{F}$) is also in P.
II. **Function set $\mathcal{F}$ violates the condition**. Then (a) Holant($\mathcal{F}$) is #P-hard or (b) #CSP($\mathcal{F}$) is #P-hard but Holant($\mathcal{F}$) is tractable.

First we discuss Part II. We can prove #P-hardness directly, or further induce an orthogonal holographic transformation. After performing the transformation, we have to handle real-valued functions. Luckily, we can even prove a dichotomy theorem for a family of *complex-valued* Holant problems (Theorem 9). And towards this theorem, we prove a lemma (Lemma 6) on how to "extract" a function from its tensor powers. The proof is non-constructive and the idea can simplify some existing proofs. For example, it can be shown directly that the two problems #CSP$^d(\mathcal{F} \cup \{[1,0]^{\otimes d}, [0,1]^{\otimes d}\})$ and #CSP$^d(\mathcal{F} \cup \{[1,0],[0,1]\})$ in [28] are equivalent under polynomial-time Turing reduction.

Now consider Part I. It can be derived that $\mathcal{F}$ is of affine type or $\mathcal{F}$ is of product type, exactly the criterion given by Dyer, Goldberg and Jerrum [19]. Dichotomies for #CSP over general domains [1, 23, 3, 8] are very different from those over the Boolean domain [18, 19]. Our proof builds a connection between them.

The Block-rank-one condition is a little conceptual. To obtain the structure of $\mathcal{F}$, we introduce an equivalent notion, called *balance*, for Holant problems (see subsection 7.2). The equivalence is simply built on the concept of *vector representation* in [8], which was used to design a polynomial-time algorithm for #CSP. Back to non-negative #CSP, we find that actually the notions of weak balance and balance (different from our version for Holant) in [8] are equivalent, without assuming FP $\neq$ #P. Therefore, to decide the complexity of a problem #CSP($\mathcal{F}$), we only need to decide whether $\mathcal{F}$ is of weak balance.

2 Preliminaries

2.1 Functions and Signatures

Let $\mathbb{C}$ and $\mathbb{R}_+$ denote the set of algebraic complex numbers and the set of algebraic non-negative real numbers, respectively. Throughout this paper, we refer to them simply as complex and non-negative numbers.

Given a function $f : \{0,1\}^n \to \mathbb{C}$, we will often write it as a vector of dimension 2^n whose entries are the function values, indexed by $\mathbf{x} \in \{0,1\}^n$ lexicographically. This vector is called a *signature*. If the values of an n-ary function only depend on the Hamming weights of inputs, then the function is called *symmetric* and can be expressed as $[f_0, f_1, ..., f_n]$ where f_k is the function value for inputs of Hamming weight k. For example, the ternary logic OR function has the signature $[0,1,1,1]$.

Generally, given a function f of arity n, we can express it as a $2^r \times 2^{n-r}$ matrix $(1 \leq r \leq n)$, denoted by $M_{[r]}(f)$. The rows and columns are indexed by $\mathbf{x} \in \{0,1\}^r$ and $\mathbf{y} \in \{0,1\}^{n-r}$ respectively, and $f(\mathbf{x}, \mathbf{y})$ is the $(\mathbf{x}, \mathbf{y})^{\text{th}}$ entry of the matrix. And the matrices $\{M_{[r]}(f) \,|\, r \in [n]\}$ are called the *signature matrices* of f. When the integer r is clear from the context, we simply write M_f.

In most cases, if not confused, we identify functions, signatures and signature matrices. But in section 7, we shall distinguish a function from its matrix representations.

Given an n-ary function f and a permutation π on $[n]$, we define the function f_π : For $x_1, x_2, ..., x_n \in \{0,1\}$, $f_\pi(x_1, x_2, ..., x_n) = f(x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(n)})$.

A function F is *reducible* if F_π is a tensor product of two functions (of arity ≥ 1) for some permutation π. Otherwise F is called *irreducible*. A function is called *degenerate* if it is a tensor product of some unary functions. Otherwise we call it *non-degenerate*.

Given a positive integer k, we use $=_k$ to denote the k-ary *equality* function $[1, 0, ..., 0, 1]$. And we use $\neq_2$ to denote the binary *disequality* function $[0, 1, 0]$.

In the following, we define three classes of complex-valued functions. Let $\mathcal{T}$ denote the set of functions that can be expressed as a tensor product of functions of arity at most 2.

The *support* of an n-ary function f, denoted by $\text{supp}(f)$, is the set $\{\mathbf{x} \in \mathbb{Z}_2^n \mid f(\mathbf{x}) \neq 0\}$. A Boolean relation is *affine* if it is the set of solutions to a system of linear equations over the field $\mathbb{Z}_2$. We say that f has affine support if its support is affine.

▶ **Definition 1.** A function f of arity n is *affine* if its support is affine and there is a constant $\lambda \in \mathbb{C}$ such that for all $\mathbf{x} \in \text{supp}(f)$, $f(\mathbf{x}) = \lambda \cdot i^{Q(\mathbf{x})}$, where $i = \sqrt{-1}$ and Q is a quadratic polynomial $Q(x_1, ..., x_n) = \sum_{i=1}^{n} a_i x_i^2 + 2\sum_{1 \leq i < j \leq n} b_{ij} x_i x_j$ with $a_i \in \mathbb{Z}_4$ and $b_{ij} \in \{0, 1\}$. We use $\mathcal{A}$ to denote the set of all affine functions.

▶ **Definition 2.** A function f is of *product type* if it can be expressed as a product of unary functions, binary functions of the form $=_2$ and $\neq_2$ (on not necessarily disjoint subsets of variables). We use $\mathcal{P}$ to denote the set of all functions of product type.

2.2 Holographic Reductions

To introduce the holographic reductions, we define bipartite Holant problems. $\text{Holant}(\mathcal{F} \mid \mathcal{G})$ denotes the Holant problem on bipartite graphs $H = (U, V, E)$ where each vertex in U (V) is assigned a function from $\mathcal{F}$ ($\mathcal{G}$). A Holant problem $\text{Holant}(\mathcal{F})$ can seen as the bipartite problem $\text{Holant}(=_2 \mid \mathcal{F})$.

Let T be a 2×2 matrix and let $\mathcal{F}$ be a function set. Whenever we write $T\mathcal{F}$, the functions in $\mathcal{F}$ are viewed as column vectors and, $T\mathcal{F} = \{T^{\otimes n} f \mid f \in \mathcal{F} \text{ and } n = \text{arity}(f)\}$. Similarly, $\mathcal{F}T = \{fT^{\otimes n} \mid f \in \mathcal{F} \text{ and } n = \text{arity}(f)\}$ where the functions in $\mathcal{F}$ are expressed as row vectors.

Let T be a matrix in $GL_2(\mathbb{C})$. We say there is a *holographic reduction* defined by T from $\text{Holant}(\mathcal{F} \mid \mathcal{G})$ to $\text{Holant}(\mathcal{F}' \mid \mathcal{G}')$, if $\mathcal{F}T \subseteq \mathcal{F}'$ and $T^{-1}\mathcal{G} \subseteq \mathcal{G}'$. The holographic reduction maps a signature grid $\Omega = (G, \mathcal{F} \mid \mathcal{G}, \pi)$ to $\Omega' = (G, \mathcal{F}' \mid \mathcal{G}', \pi')$: For each vertex v of G, π' assigns the function $f_v T$ or $T^{-1} f_v$ to v, depending on which part v belongs to.

▶ **Theorem 3** (Valiant's Holant Theorem [32])**.** *Let T be any matrix in $GL_2(\mathbb{C})$. Suppose that the holographic reduction defined by T maps a signature grid Ω to Ω'. Then* $\text{Holant}_\Omega = \text{Holant}_{\Omega'}$.

We will use $\leq_{\mathsf{T}}$ to denote polynomial-time Turing reductions and use $\equiv_{\mathsf{T}}$ to denote the equivalence relation under polynomial-time Turing reductions.

▶ **Theorem 4.** *Let F be a function set and let H be an orthogonal matrix ($H^{\mathsf{T}}H = I$). Then* $\text{Holant}(H\mathcal{F}) \equiv_{\mathsf{T}} \text{Holant}(\mathcal{F})$.

2.3 Realizability

Let $\mathcal{F}$ be a set of functions. An $\mathcal{F}$-gate [15] Γ is a tuple $(G, \mathcal{F}, \pi)$ where $G = (V, E, D)$ is a graph with regular edges E and some dangling edges D. Other than these dangling edges, the gate Γ is the same as a signature grid: π maps each vertex $v \in V$ to some function $f_v \in \mathcal{F}$ and it incident edges (including the dangling ones) to the input variables of f_v. We denote the edges in E by $1, 2, ..., m$ and the dangling edges in D by $m+1, m+2, ..., m+n$. Then we can define a function f for Γ:

$$f(y_1, y_2, ..., y_n) = \sum_{x_1, x_2, ..., x_m \in \{0,1\}} F(x_1, x_2, ..., x_m, y_1, y_2, ..., y_n)$$

where $(y_1, y_2, ..., y_n) \in \{0,1\}^n$ is an assignment on the dangling edges and $F(\mathbf{x}, \mathbf{y})$ denotes the product of evaluations at all vertices of V. We say the function f is *realizable* from the function set $\mathcal{F}$. We use $\mathsf{S}(\mathcal{F})$ to denote the set of functions realizable from $\mathcal{F}$.

Given a function f, we use $f^{x_i=c}$ to denote the function obtained by pinning the ith input variable of f to $c \in \{0,1\}$.

2.4 Weighted Counting CSP

Let $\mathcal{F}$ be a set of complex-valued functions. Then the problem #CSP($\mathcal{F}$) is defined as follows. An input instance I of the problem consists of a finite set of variables $V = \{x_1, ..., x_n\}$ and a finite set of constraints $\{C_1, ..., C_m\}$. Each C_i has the form $(F_i, \mathbf{x}_i)$ where $F_i \in \mathcal{F}$ and $\mathbf{x}_i$ is a tuple of (not necessarily distinct) variables from V. The instance I *defines* a function F_I over $\mathbf{x} = (x_1, ..., x_n) \in \{0,1\}^n$: $F_I(\mathbf{x}) = \prod_{i=1}^m F_i(\mathbf{x}_i)$ for $\mathbf{x} \in \{0,1\}^n$. The output is the sum: $Z(I) = \sum_{\mathbf{x} \in \{0,1\}^n} F_I(\mathbf{x})$.

Holant problems are indeed read-twice #CSPs. Given a signature grid, we assume that the numbering of its vertices and edges is also given. If these edges are viewed as variables, then the signature grid is a #CSP instance where every variable appears exactly twice. So we also say that a signature grid defines a function. And the concept of realizability can be defined in the CSP language.

Cai, Lu and Xia [17] proved a dichotomy for complex-weighted #CSP over the Boolean domain.

▶ **Theorem 5** ([17]). *Let $\mathcal{F}$ be a set of complex-valued functions. Then the problem* #CSP($\mathcal{F}$) *is computable in polynomial time if $\mathcal{F} \subseteq \mathcal{A}$ or $\mathcal{F} \subseteq \mathcal{P}$. Otherwise* #CSP($\mathcal{F}$) *is* #P*-hard.*

3 Decomposition

In Holant problems, sometimes we are able to realize a function $F = f \otimes g$, but do not know how to realize the function f directly, which can be technically beneficial. Fortunately, under certain conditions, if F is realizable, then we may assume that f is freely available.

In this section, we prefer to prove the lemmas in the CSP language. If not specified, the functions we discussed are over a fixed finite domain and take complex values.

Let m be a positive integer. We use $f^{\otimes m}$ to denote the m-th tensor power of f. $f^{\otimes m}$ can be seen as m copies of f: $f^{\otimes m}(\mathbf{x}_1, ..., \mathbf{x}_m) = f(\mathbf{x}_1) \cdots f(\mathbf{x}_m)$. Let I be a #CSP instance that contains m constraints: $(f, \mathbf{x}_1), (f, \mathbf{x}_2), ..., (f, \mathbf{x}_m)$. We replace these m tuples by one tuple $(f^{\otimes m}, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m)$ and then obtain a new instance I'. It is easy to see that $Z(I) = Z(I')$.

▶ **Lemma 6.** *For any function set $\mathcal{F}$ and function f,* Holant($\mathcal{F} \cup \{f\}$) $\leq_{\mathsf{T}}$ Holant($\mathcal{F} \cup \{f^{\otimes d}\}$) *for all $d \geq 1$.*

Proof. Impose induction on d. Let n denote the arity of f.

The base case, $d = 1$, is trivial. Now suppose that the conclusion holds for all $d < k$ ($k \geq 2$). In the problem Holant($\mathcal{F} \cup \{f^{\otimes k}\}$), we may assume that the functions $f^{\otimes (mk)}$ are freely available for integers $m > 0$. There are two cases to consider:

- There exists an instance I of Holant($\mathcal{F} \cup \{f\}$) such that $Z(I) \neq 0$ and f appears p times where $p = qk + r$ ($q \geq 0, 0 < r < k$). Let $C_1, ..., C_p$ be the p constraints that have the form $(f, \mathbf{x}_i)$. We replace the first qk constraints by one tuple $C_1' = (f^{\otimes (qk)}, \mathbf{x}_1, ..., \mathbf{x}_{qk})$, and the last r constraints by one tuple $C_2' = (f^{\otimes k}, \mathbf{x}_{qk+1}, ..., \mathbf{x}_p, \mathbf{y})$ where $\mathbf{y}$ denotes a list of new distinct variables, of length $(k - r)n$. After the substitution, we get a function

$F(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ denotes the variables of the original instance I. Every variable in $\mathbf{x}$ occurs twice, so by summing on them we can realize the following function:

$$\sum_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{x}} F_I(\mathbf{x}) f^{\otimes(k-r)}(\mathbf{y}) = Z(I) f^{\otimes(k-r)}(\mathbf{y}).$$

Because $Z(I) \neq 0$, we have $\text{Holant}(\mathcal{F} \cup \{f^{\otimes(k-r)}\}) \leq_{\mathsf{T}} \text{Holant}(\mathcal{F} \cup \{f^{\otimes k}\})$. And by the induction hypothesis, $\text{Holant}(\mathcal{F} \cup \{f\}) \leq_{\mathsf{T}} \text{Holant}(\mathcal{F} \cup \{f^{\otimes(k-r)}\})$. Therefore, the conclusion holds.

- For all I with $Z(I) \neq 0$, f appears a multiple of k times. Given an instance I of $\text{Holant}(\mathcal{F} \cup \{f\})$, we show how to compute $Z(I)$ with the help of the oracle for $\text{Holant}(\mathcal{F} \cup \{f^{\otimes k}\})$. First we check whether the number p of constraints containing f is a multiple of k. If not, we simply output 0. Otherwise we replace all such constraints by one tuple $(f^{\otimes p}, \mathbf{x})$ as in case (1), and then obtain an instance I' of $\text{Holant}(\mathcal{F} \cup \{f^{\otimes k}\})$. Clearly $Z(I) = Z(I')$, and we can compute $Z(I')$ by accessing the oracle.

In either case, there exists a polynomial-time Turing reduction. This completes the induction. ◀

Note that our proof only shows the *existence* of polynomial-time Turing reductions, but does not produce such reductions *constructively* for given function sets. Based on Lemma 6, we can prove a more general one.

▶ **Lemma 7.** *Let $\mathcal{F}$ be a set of functions, and f, g be two functions. Suppose that there exists an instance I of* $\text{Holant}(\mathcal{F} \cup \{f, g\})$ *such that $Z(I) \neq 0$, and the number of occurrences of g in I is greater than that of f. Then* $\text{Holant}(\mathcal{F} \cup \{f, f \otimes g\}) \leq_{\mathsf{T}} \text{Holant}(\mathcal{F} \cup \{f \otimes g\})$.

4 When A Non-trivial Equality Function Appears

Let $\text{Holant}^c(\mathcal{F})$ denote the problem $\text{Holant}(\mathcal{F} \cup \{[1,0],[0,1]\})$. We have the following theorem:

▶ **Theorem 8.** *Let λ be any nonzero complex number that is not a root of unity. For any set $\mathcal{F}$ of complex-valued functions,* $\text{Holant}^c(\mathcal{F} \cup \{[1,0,\lambda]\})$ *is computable in polynomial time if $\mathcal{F} \subseteq \mathcal{T}$ or $\mathcal{F} \subseteq \mathcal{P}$. Otherwise the problem is #P-hard.*

The conclusion still holds if we remove the unary functions $[1,0]$ and $[0,1]$:

▶ **Theorem 9.** *Let λ be any nonzero complex number that is not a root of unity. For any set $\mathcal{F}$ of complex-valued functions,* $\text{Holant}(\mathcal{F} \cup \{[1,0,\lambda]\})$ *is computable in polynomial time if $\mathcal{F} \subseteq \mathcal{T}$ or $\mathcal{F} \subseteq \mathcal{P}$. Otherwise the problem is #P-hard.*

Proof. We can interpolate $[1,0]^{\otimes 2}$ and $[0,1]^{\otimes 2}$ using $[1,0,\lambda]$. Then by Lemma 6, $\text{Holant}^c(\mathcal{F} \cup \{[1,0,\lambda]\}) \leq_{\mathsf{T}} \text{Holant}(\mathcal{F} \cup \{[1,0,\lambda]\})$. ◀

Intuitively, we can interpolate all functions of the form $[a,0,b]$, using the binary function $[1,0,\lambda]$. By connecting with these binary funtions, a function f may range arbitrarily. To avoid #P-hardness, the structure of the support of f must be simple enough.

5 $\mathcal{P}$-transformability

We start with some simple facts from linear algebra. Let $M = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \end{bmatrix}$ $(n \geq 2)$ be a non-negative matrix of rank 2. Then $A = MM^{\mathsf{T}} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$ satisfying $a, c > 0$. Moreover, by Cauchy-Schwarz inequality, $\det A = ac - b^2 > 0$.

▶ **Lemma 10.** *If $a \neq c$ or $b \neq 0$, then A has two distinct positive eigenvalues α and β.*

The following lemma is a simple case of the Spectral Theorem for real symmetric matrices.

▶ **Lemma 11.** *There is an orthogonal matrix H such that $HAH^{\mathsf{T}} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}$, where α and β are the eigenvalues of A.*

Let f be a non-negative binary function. If f is non-degenerate and affine, then $f = a[1,0,1]$ or $f = a[0,1,0]$ for some $a > 0$.

▶ **Lemma 12.** *Let $f = (a,b,c,d)$ be a non-negative function. Suppose that f is non-degenerate and $f \notin \mathcal{A}$. Then for any function set $\mathcal{F}$ with $f \in \mathsf{S}(\mathcal{F})$, $\mathrm{Holant}(\mathcal{F})$ is #P-hard or $\mathcal{F} \subseteq \mathcal{T}$ or $\mathcal{F} \subseteq H\mathcal{P}$ for some orthogonal matrix H.*

Proof. Since $f \in \mathsf{S}(\mathcal{F})$, the symmetric matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} a^2+b^2 & ac+bd \\ ac+bd & c^2+d^2 \end{bmatrix}$$

is also realizable. Because f is non-degenerate, $a^2+b^2, c^2+d^2 > 0$ and $ac+bd \geq 0$. We claim that $ac+bd \neq 0$ or $a^2+b^2 \neq c^2+d^2$. Suppose $ac+bd = 0$, then $ac = bd = 0$ since f is non-negative. So $f = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$ or $f = \begin{bmatrix} 0 & b \\ c & 0 \end{bmatrix}$. In both cases, as $f \notin \mathcal{A}$, $a^2+b^2 \neq c^2+d^2$.

By Lemma 10 and Lemma 11, there is some orthogonal matrix H such that $HAH^{\mathsf{T}} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}$, where α and β are the two distinct positive eigenvalues of A. Now we perform the transformation H and obtain the following equivalence:

$$\mathrm{Holant}(\{[\alpha,0,\beta]\} \cup H\mathcal{F}) \equiv_{\mathsf{T}} \mathrm{Holant}(\{A\} \cup \mathcal{F}) \equiv_{\mathsf{T}} \mathrm{Holant}(\mathcal{F}).$$

The latter equivalence follows from the fact $A \in \mathsf{S}(f) \subseteq \mathsf{S}(\mathcal{F})$. β/α is nonzero and not a root of unity, so if $H\mathcal{F} \not\subseteq \mathcal{T}$ and $H\mathcal{F} \not\subseteq \mathcal{P}$, the problem is #P-hard by Theorem 9. ◀

6 On Special Functions of Arity 4

In this section, we consider some special functions of arity 4, and complete the preparation for the hardness part of our dichotomy.

▶ **Lemma 13.** *Let f be a function of arity 4, whose signature matrix has the form*

$$M_f = \begin{bmatrix} f_{0000} & f_{0001} & f_{0010} & f_{0011} \\ f_{0100} & f_{0101} & f_{0110} & f_{0111} \\ f_{1000} & f_{1001} & f_{1010} & f_{1011} \\ f_{1100} & f_{1101} & f_{1110} & f_{1111} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & b & c & 0 \\ 0 & c & b & 0 \\ a & 0 & 0 & 1 \end{bmatrix}$$

where $a,b,c \geq 0$ and at least two of them are positive. Then $\mathrm{Holant}(f)$ is #P-hard if $f \neq [1,0,1,0,1]$.

We prove a dichotomy for function sets that contain certain functions of arity 4.

▶ **Lemma 14.** *Let f be a non-negative function of arity 4. And $\begin{bmatrix} f_{0000} & f_{0011} \\ f_{1100} & f_{1111} \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$ where $b \neq 0$ and $ac > b^2$. Then for any function set $\mathcal{F}$ containing f, $\mathrm{Holant}(\mathcal{F})$ is #P-hard or $\mathcal{F} \subseteq \mathcal{T}$ or $\mathcal{F} \subseteq H\mathcal{P}$ for some orthogonal matrix H.*

Proof. We can show that $\mathrm{Holant}(\mathcal{F})$ is #P-hard by Theorem 5 and Lemma 13, or there is some non-negative binary function $f \notin \mathcal{A} \cup \mathcal{P}$ such that $\mathrm{Holant}(\mathcal{F} \cup \{f\}) \leq_{\mathsf{T}} \mathrm{Holant}(\mathcal{F})$. Then the conclusion follows from Lemma 12. ◀

7 The Dichotomy

7.1 The Block-rank-one Condition Captures the Dichotomy

Given a function f of arity n, we use $f^{[t]}$, for each $t \in [n]$, to denote the function

$$f^{[t]}(x_1, ..., x_t) = \sum_{x_{t+1},...,x_n \in \{0,1\}} f(x_1, ..., x_t, x_{t+1}, ..., x_n).$$

Recall that Holant problems are read-twice #CSPs and every #CSP instance *defines* a function (subsection 2.4). We adopt the notation in [7], defining the following set of functions for a given $\mathcal{F}$:

$$\mathcal{W}_{\mathcal{F}} = \{F^{[t]} \mid F \text{ is a function defined by an instance of Holant}(\mathcal{F}) \text{ and } 1 \le t \le \text{arity of } F\}.$$

Note that the functions in $\mathcal{W}_{\mathcal{F}}$ are not necessarily realizable from $\mathcal{F}$. The following two lemmas show how $\mathcal{W}_{\mathcal{F}}$ and $\mathsf{S}(\mathcal{F})$ are related:

▶ **Lemma 15.** *Let $f \in \mathcal{W}_{\mathcal{F}}$ be a function of arity n. Then there is a function $g \in \mathsf{S}(\mathcal{F})$ of arity $2n$, such that for all $x_1, x_2, ..., x_n \in \{0, 1\}$, $f(x_1, x_2, ..., x_n) = g(x_1, x_1, x_2, x_2, ..., x_n, x_n)$.*

▶ **Lemma 16.** *For $f \in \mathsf{S}(\mathcal{F})$, $f^2 \in \mathcal{W}_{\mathcal{F}}$.*

Let M be a non-negative matrix. We say M is *block-rank-one* if every two rows of it are linearly dependent or orthogonal. Given a non-negative function f of arity n, we say f is *block-rank-one* if either $n = 1$ or the matrix $M_{[n-1]}(f)$ is block-rank-one.

Now we impose a condition on $\mathcal{W}_{\mathcal{F}}$:

Block-rank-one: All functions in $\mathcal{W}_{\mathcal{F}}$ are block-rank-one.

We can classify those function sets that do not satisfy this condition:

▶ **Lemma 17.** *Let $\mathcal{F}$ be a set of non-negative functions. If $\mathcal{F}$ does not satisfy the Block-rank-one condition, then* Holant$(\mathcal{F})$ *is #P-hard or $\mathcal{F} \subseteq \mathcal{T}$ or $\mathcal{F} \subseteq H\mathcal{P}$ for some orthogonal matrix H.*

Proof. Let $f \in \mathcal{W}_{\mathcal{F}}$ be a function of arity n. Then by Lemma 15, there is a function $g \in \mathsf{S}(\mathcal{F})$ of arity $2n$, such that for all $x_1, x_2, ..., x_n \in \{0, 1\}$, $f(x_1, x_2, ..., x_n) = g(x_1, x_1, x_2, x_2, ..., x_n, x_n)$.

Now suppose that f is not block-rank-one. By definition, $n \ge 2$ and the two columns of $M_{[n-1]}(f)$ are linearly independent but not orthogonal. Then the first and the last columns of the matrix $M = M_{[2n-2]}(g)$, $g^{x_{2n-1}=x_{2n}=0}$ and $g^{x_{2n-1}=x_{2n}=1}$, are also linearly independent but not orthogonal. Let h denote the 4×4 matrix $M^{\mathsf{T}}M$. Then $h_{0011} = h_{1100} > 0$ and $h_{0000}h_{1111} > h_{0011}^2$. Since $g \in \mathsf{S}(\mathcal{F})$, h is also realizable. Thus Holant$(\mathcal{F} \cup \{h\}) \le_{\mathsf{T}}$ Holant$(\mathcal{F})$. By Lemma 14, Holant$(\mathcal{F})$ is #P-hard or $\mathcal{F} \subseteq \mathcal{T}$ or $\mathcal{F} \subseteq H\mathcal{P}$ for some orthogonal H. ◀

Surprisingly, the Block-rank-one condition has *captured* the dichotomy. We have the crucial lemma below:

▶ **Lemma 18.** *Let $\mathcal{F}$ be a set of non-negative functions. If $\mathcal{F}$ satisfies the Block-rank-one condition, then $\mathcal{F} \subseteq \mathcal{A}$ or $\mathcal{F} \subseteq \mathcal{P}$.*

Therefore, if $\mathcal{F}$ satisfies the Block-rank-one condition, then Holant$(\mathcal{F})$ is in polynomial time. So our dichotomy is quite simple and it is decidable in polynomial time [12]:

▶ **Theorem 19.** *Let $\mathcal{F}$ be a set of non-negative functions. The problem* Holant($\mathcal{F}$) *is computable in polynomial time if $\mathcal{F}$ satisfies one of the following three conditions:*

- $\mathcal{F} \subseteq \mathcal{T}$;
- $\mathcal{F} \subseteq \mathcal{A}$;
- $\mathcal{F} \subseteq H\mathcal{P}$ *for some real orthogonal matrix* H.

Otherwise Holant($\mathcal{F}$) *is* #P*-hard.*

The remaining is to prove Lemma 18. To obtain the structure of $\mathcal{F}$, it is more convenient to consider directly the set $\mathcal{F}$ and the functions realizable from it. So in the next subsection, we will introduce a notion equivalent to the Block-rank-one condition. This notion restricts the function set $\mathsf{S}(\mathcal{F})$.

7.2 Balance

We define the notion of *balance* for non-negative Holant problems. The notion was introduced for non-negative #CSP by Cai, Chen and Lu [8].

▶ **Definition 20** (Balance)**.** Let $\mathcal{F}$ be a set of non-negative functions. $\mathcal{F}$ is called *balanced* if for any function $f \in \mathsf{S}(\mathcal{F})$, every signature matrix in $\{M_{[r]}(f) \mid 1 \leq r \leq \text{arity}(f)\}$ is block-rank-one. A non-negative function f is *balanced* if the set $\{f\}$ is balanced.

Note that in the definition above, when $r = \text{arity}(f)$, the matrix $M_{[r]}(f)$ is a column vector and hence trivially block-rank-one.

Balanced sets satisfy the Block-rank-one condition. Generally, we have the following lemma.

▶ **Lemma 21.** *Let $\mathcal{F}$ be a set of non-negative functions. Suppose that $\mathcal{F}$ is balanced. Then for any $f \in \mathcal{W}_{\mathcal{F}}$, every matrix in $\{M_{[r]}(f) \mid 1 \leq r \leq \text{arity}(f)\}$ is block-rank-one.*

Proof. Let $f \in \mathcal{W}_{\mathcal{F}}$ be a function of arity n. Then by Lemma 15, there exists a function $g \in \mathsf{S}(\mathcal{F})$ of arity $2n$, such that for all $x_1, x_2, ..., x_n \in \{0, 1\}$,

$$f(x_1, x_2, ..., x_n) = g(x_1, x_1, x_2, x_2, ..., x_n, x_n).$$

Therefore, for any $r \in [n]$, $M_{[r]}(f)$ is a submatrix of $M_{[2r]}(g)$. Because $\mathcal{F}$ is balanced, $M_{[2r]}(g)$ is block-rank-one. Hence so is $M_{[r]}(f)$. ◀

Let f be a non-negative function of arity n. And let $s_1, ..., s_n$ be n non-negative unary functions. We call $(s_1, ..., s_n)$ a *vector representation* of f if for all $\mathbf{x} \in \{0, 1\}^n$, either $f(\mathbf{x}) = 0$ or $f(\mathbf{x}) = s_1(x_1) \cdots s_n(x_n)$.

▶ **Lemma 22** ([8])**.** *Let f be a non-negative function of arity n. If $f^{[t]}$ is block-rank-one for all $t \in [n]$, then f has a vector representation.*

▶ **Lemma 23.** *Let $\mathcal{F}$ be a set of non-negative functions that satisfies the Block-rank-one condition. Then every function in $\mathsf{S}(\mathcal{F})$ has a vector representation.*

Proof. Let f be a function in $\mathsf{S}(\mathcal{F})$ of arity n. By Lemma 16, $f^2 \in \mathcal{W}_{\mathcal{F}}$. Then f^2 has a vector representation $(s_1, ..., s_n)$ by Lemma 22. Let $(s'_1, ..., s'_n)$ be n non-negative unary functions such that for all $i \in [n]$, $s'_i(a) = \sqrt{s_i(a)}$ for $a \in \{0, 1\}$. Then $(s'_1, ..., s'_n)$ is a vector representation of the function f. ◀

Now we are able to prove the equivalence between the notion of balance and the Block-rank-one condition.

▶ **Lemma 24.** *Let $\mathcal{F}$ be a set of non-negative functions. $\mathcal{F}$ is balanced if and only if $\mathcal{F}$ satisfies the Block-rank-one condition.*

Proof. The necessity follows directly from Lemma 21. We only need to show the sufficiency.

Let f be an n-ary function in $\mathsf{S}(\mathcal{F})$, with $n \geq 2$. And suppose that $M = M_{[r]}(f)$ is not block-rank-one for some $r \in [n]$. Then there exist two rows of M, indexed by some $\mathbf{x}, \mathbf{y} \in \{0,1\}^r$, which are linearly independent but not orthogonal. So we can realize a signature $g = MM^{\mathsf{T}}$. Its submatrix

$$h = \begin{bmatrix} g(\mathbf{x},\mathbf{x}) & g(\mathbf{x},\mathbf{y}) \\ g(\mathbf{y},\mathbf{x}) & g(\mathbf{y},\mathbf{y}) \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

is of full rank and $a, b, c > 0$. But by Lemma 23, g has a vector representation $(s_1, ..., s_{2r})$, such that for all $\mathbf{u} \in \mathrm{supp}(g)$, $g(\mathbf{u}) = s_1(u_1) \cdots s_{2r}(u_{2r})$. Let $s = s_1 \otimes \cdots \otimes s_r$ and $t = s_{r+1} \otimes \cdots \otimes s_{2r}$. Then

$$h = \begin{bmatrix} s(\mathbf{x})t(\mathbf{x}) & s(\mathbf{x})t(\mathbf{y}) \\ s(\mathbf{y})t(\mathbf{x}) & s(\mathbf{y})t(\mathbf{y}) \end{bmatrix},$$

which is singular. A contradiction. ◀

Having shown the equivalence, we turn to consider some properties of balanced sets. There are two basic facts about balance. Later we will often use them but without explicit reference.

▶ **Lemma 25.** *If $\mathcal{F} \subseteq \mathcal{G}$ and $\mathcal{G}$ is balanced, then $\mathcal{F}$ is also balanced.*

▶ **Lemma 26.** *If $f \in \mathsf{S}(\mathcal{F})$ and $\mathcal{F}$ is balanced, then $\mathcal{F} \cup \{f\}$ is also balanced.*

In Boolean #CSP, the two unary functions $[1, 0]$ and $[0, 1]$ can be simulated [19]. And the function $[1, 1]$ is the unary equality function, which is freely available. These unary functions make it more convenient to construct certain functions. But in Holant problems, generally we do not know how to realize or simulate them. Fortunately, we can circumvent this difficulty by the lemma below. It follows from Lemma 28 and Lemma 30.

▶ **Lemma 27.** *If $\mathcal{F}$ is balanced, then the set $\mathcal{F} \cup \{[1, 0], [0, 1], [1, 1]\}$ is balanced.*

▶ **Lemma 28.** *If $\mathcal{F}$ is balanced, then $\mathcal{F} \cup \{[1, 0], [0, 1]\}$ is balanced.*

▶ **Lemma 29.** *Suppose that $\mathcal{F}$ is a balanced set of non-negative functions. Let f be an n-ary function in $\mathsf{S}(\mathcal{F})$ and let F denote the function f^2. Then for each $t \in [n]$, there exists a constant $\lambda_t > 0$ such that $F^{[t]} = \lambda_t (f^{[t]})^2$.*

Proof. Impose induction on t. The base case $t = n$ is trivial where $\lambda_n = 1$.

Suppose that $F^{[t]} = \lambda_t (f^{[t]})^2$ for $t = k + 1 \leq n$. Consider the case $t = k$. For all $\mathbf{x} \in \{0,1\}^k$,

$$F^{[k]}(\mathbf{x}) = F^{[k+1]}(\mathbf{x}, 0) + F^{[k+1]}(\mathbf{x}, 1) = \lambda_{k+1} \left[\left(f^{[k+1]}(\mathbf{x}, 0) \right)^2 + \left(f^{[k+1]}(\mathbf{x}, 1) \right)^2 \right].$$

Note that the function $F^{[k+1]} \in \mathcal{W}_{\mathcal{F}}$ since $F = f^2 \in \mathcal{W}_{\mathcal{F}}$. Because $\mathcal{F}$ is balanced, $F^{[k+1]}$ is block-rank-one by Lemma 24. Thus the function $f^{[k+1]} = \sqrt{F^{[k+1]}/\lambda_{k+1}}$ is also block-rank-one, which implies that the two column vectors of the matrix $M_{[k]}(f^{[k+1]})$, denoted by $\mathbf{v}_0$ and $\mathbf{v}_1$, are orthogonal or linearly dependent:

- $\mathbf{v}_0$ and $\mathbf{v}_1$ are orthogonal. Then for all $\mathbf{x} \in \{0,1\}^k$,

$$F^{[k]}(\mathbf{x}) = \lambda_{k+1}\left[\left(f^{[k+1]}(\mathbf{x},0)\right)^2 + \left(f^{[k+1]}(\mathbf{x},1)\right)^2\right]$$
$$= \lambda_{k+1}\left(f^{[k+1]}(\mathbf{x},0) + f^{[k+1]}(\mathbf{x},1)\right)^2 = \lambda_{k+1}\left(f^{[k]}(\mathbf{x})\right)^2.$$

- $\mathbf{v}_0$ and $\mathbf{v}_1$ are linearly dependent. Without loss of generality, we assume that $\mathbf{v}_1 = \lambda\mathbf{v}_0$ for some $\lambda \geq 0$. Then for all $\mathbf{x} \in \{0,1\}^k$,

$$F^{[k]}(\mathbf{x}) = \lambda_{k+1}(1+\lambda^2)\left(f^{[k+1]}(\mathbf{x},0)\right)^2 = \lambda_{k+1}\frac{1+\lambda^2}{(1+\lambda)^2}\left(f^{[k]}(\mathbf{x})\right)^2.$$

In either case, the conclusion holds. This completes the induction. ◀

▶ **Lemma 30.** *If $\mathcal{F}$ is balanced, then $\mathcal{F} \cup \{[1,1]\}$ is balanced.*

Proof. Suppose that $[1,1] \notin \mathsf{S}(\mathcal{F})$, otherwise we are done. Let g be an n-ary function in $\mathsf{S}(\mathcal{F} \cup \{[1,1]\})$. We need to show that all the matrices in $\{M_{[r]}(g) \mid 1 \leq r \leq \text{arity}(g)\}$ are block-rank-one.

Let Γ denote the gate that realizes g. If there is an isolated vertex with a dangling edge in Γ, assigned the function $[1,1]$, then we remove this vertex; If there are two adjacent vertices, both assigned the function $[1,1]$, then we delete the pair. Repeat removing until no such vertices. Finally we obtain a new gate Γ'. If Γ' has no dangling edges, then we are done. Suppose not. Let h denote the function that Γ' realizes. And for all $x_1, ..., x_n \in \{0,1\}$, $g(x_1, ..., x_n) = 2^s h(x_{i_1}, ..., x_{i_t})$ where $1 \leq i_1 < \cdots < i_t \leq n$ and s denotes the number of pairs we delete. It suffices to prove that the signature matrices of h are all block-rank-one.

Note that $h = f^{[t]}$ for some $f \in \mathsf{S}(\mathcal{F})$ and $1 \leq t \leq \text{arity}(f)$. Let F denote the function f^2. Then by Lemma 29, there is a constant $\lambda_t > 0$ such that $F^{[t]} = \lambda_t (f^{[t]})^2$. Therefore, for any $r \in [t]$, the two matrices $M_{[r]}(f^{[t]})$ and $M_{[r]}(F^{[t]})$ are both block-rank-one or neither. Since $F^{[t]} \in \mathcal{W}_{\mathcal{F}}$, all of its signature matrices are block-rank-one by Lemma 21. Thus every matrix in $\{M_{[r]}(f^{[t]}) \mid 1 \leq r \leq t\}$ is block-rank-one. ◀

With these unary functions, we are able to prove two more lemmas:

▶ **Lemma 31.** *Let $\mathcal{F}$ be a set of non-negative functions and let $g = [1,0,1,0]$. If $\mathcal{F} \cup \{g\}$ is balanced, then $\mathcal{F} \subseteq \mathcal{A}$.*

▶ **Lemma 32.** *Let $\mathcal{F}$ be a set of non-negative functions and let $g = [a,0,...,0,b]$ be a general equality function where $\text{arity}(g) \geq 3$ and $a, b > 0$. If $\mathcal{F} \cup \{g\}$ is balanced, then $\mathcal{F} \subseteq \mathcal{A}$ or $\mathcal{F} \subseteq \mathcal{P}$.*

7.3 Proof Sketch of Lemma 18

Suppose that a function set $\mathcal{F}$ satisfies the Block-rank-one condition. Then the set $\mathcal{G} = \mathcal{F} \cup \{[1,0],[0,1],[1,1]\}$ is balanced. So it suffices to prove that $\mathcal{G} \subseteq \mathcal{A}$ or $\mathcal{G} \subseteq \mathcal{P}$.

First we consider the case $\mathcal{G} \subseteq \mathcal{T}$. In this case, every nondegenerate binary function in $\mathsf{S}(\mathcal{G})$ has the form $[a,0,b]$ or $(0,a,b,0)$. Thus all of them are of product type. Since the set $\mathcal{P}$ is closed under tensor product, $\mathcal{G} \subseteq \mathcal{P}$.

Now suppose that $\mathcal{G} \not\subseteq \mathcal{T}$. Then there is an irreducible function $f \in \mathsf{S}(\mathcal{G})$ of arity $n \geq 3$. For $1 \leq i < j \leq n$ and $a, b \in \{0,1\}$, we use f_{ij}^{ab} denote the column vector $M_{[n-2]}(f^{x_i=a, x_j=b})$. And we define the $2^{n-2} \times 2^2$ matrices $M_{ij} = (f_{ij}^{00}, f_{ij}^{01}, f_{ij}^{10}, f_{ij}^{11})$.

Since f is irreducible and $\mathcal{G}$ is balanced, any two elements of the support of f differ at two or more bits. Thus we have:

$$\langle f_{ij}^{00}, f_{ij}^{01} \rangle = 0, \ \langle f_{ij}^{00}, f_{ij}^{10} \rangle = 0,$$
$$\langle f_{ij}^{11}, f_{ij}^{01} \rangle = 0, \ \langle f_{ij}^{11}, f_{ij}^{10} \rangle = 0,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Therefore, for every pair (i, j), the 4×4 matrix $B_{ij} = (M_{ij})^{\mathsf{T}} M_{ij}$ has the form

$$\begin{bmatrix} a & 0 & 0 & b \\ 0 & x & y & 0 \\ 0 & y & z & 0 \\ b & 0 & 0 & c \end{bmatrix}.$$

By Cauchy-Schwarz inequality, $ac \geq b^2$ and $xz \geq y^2$. If for all $1 \leq i < j \leq n$, B_{ij} is diagonal, then there exists a function $g = [a, 0, ..., 0, b] \in \mathsf{S}(\mathcal{G})$ where arity$(g) \geq 3$ and $a, b > 0$. If some B_{ij} is not diagonal, then $B_{ij} = a[1, 0, 1, 0, 1]$ for some $a > 0$ due to the balance of $\mathcal{G}$. In this case, we can further realize the function $a[1, 0, 1, 0]$. According to Lemma 32 or Lemma 31, $\mathcal{G} \subseteq \mathcal{A}$ or $\mathcal{G} \subseteq \mathcal{P}$.

8 Conclusion

To determine the complexity of a problem Holant$(\mathcal{F})$, the proofs of previous Holant dichotomies often start with a non-trivial function in $\mathcal{F}$. This works well for symmetric functions, but the structure of an asymmetric one can be very intricate. In [16], we have already seen that asymmetry poses great challenges in arity reduction and gadget construction, even assuming the presence of all unary functions. In fact, similar difficulty arises on higher domains, where it is tough to obtain an explicit dichotomy. The #CSP dichotomies over general domains [23, 8, 7] are more abstract than those over the Boolean domain, but they offer great insights into sum-of-product computation. Inspired by them, we introduce the Block-rank-one condition for Holant problems, which leads to a clear classification. At the beginning of our work, we were not sure whether the condition is sufficient for tractability. Lemma 24 and Lemma 27 make it possible to absorb the results in [19] and reach the destination.

Acknowledgements. The authors are grateful to Jin-Yi Cai for his careful reading of an earlier version of this paper.

References

1 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34, 2013.
2 Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Inf. Comput.*, 205(5):651–678, 2007.
3 Andrei A. Bulatov, Martin E. Dyer, Leslie A. Goldberg, Markus Jalsenius, Mark Jerrum, and David Richerby. The complexity of weighted and unweighted #CSP. *J. Comput. Syst. Sci.*, 78(2):681–688, 2012.
4 Andrei A. Bulatov, Martin E. Dyer, Leslie A. Goldberg, Markus Jalsenius, and David Richerby. The complexity of weighted Boolean #CSP with mixed signs. *Theor. Comput. Sci.*, 410(38-40):3949–3961, 2009.

5 Andrei A. Bulatov and Martin Grohe. The complexity of partition functions. *Theor. Comput. Sci.*, 348(2):148–186, 2005.

6 Jin-Yi Cai and Xi Chen. A decidable dichotomy theorem on directed graph homomorphisms with non-negative weights. In *Proceedings of 51th Annual IEEE Symposium on Foundations of Computer Science*, pages 437–446, 2010.

7 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. In *Proceedings of the 44th Symposium on Theory of Computing*, pages 909–920, 2012.

8 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Non-negatively weighted #CSP: An effective complexity dichotomy. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity*, pages 45–54, 2011.

9 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Graph homomorphisms with complex values: A dichotomy theorem. *SIAM J. Comput.*, 42(3):934–1029, 2013.

10 Jin-Yi Cai, Zhiguo Fu, Heng Guo, and Tyson Williams. A Holant dichotomy: Is the FKT algorithm universal? In *IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1259–1276, 2015.

11 Jin-Yi Cai, Heng Guo, and Tyson Williams. A complete dichotomy rises from the capture of vanishing signatures: Extended abstract. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 635–644, 2013.

12 Jin-Yi Cai, Heng Guo, and Tyson Williams. Holographic algorithms beyond matchgates. In *Proceedings of ICALP*, pages 271–282, 2014.

13 Jin-Yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and back: Dichotomy for Holant^c problems. *Algorithmica*, 64(3):511–533, 2012.

14 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holant problems and counting CSP. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 715–724, 2009.

15 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Computational complexity of Holant problems. *SIAM J. Comput.*, 40(4):1101–1132, 2011.

16 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Dichotomy for Holant* problems of Boolean domain. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1714–1728, 2011.

17 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted Boolean #CSP. *J. Comput. Syst. Sci.*, 80(1):217–236, 2014.

18 Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996.

19 Martin E. Dyer, Leslie A. Goldberg, and Mark Jerrum. The complexity of weighted Boolean #CSP. *SIAM J. Comput.*, 38(5):1970–1986, 2009.

20 Martin E. Dyer, Leslie A. Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54(6), 2007.

21 Martin E. Dyer and Catherine S. Greenhill. The complexity of counting graph homomorphisms. *Random Struct. Algorithms*, 17(3-4):260–289, 2000.

22 Martin E. Dyer and David Richerby. On the complexity of #CSP. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 725–734, 2010.

23 Martin E. Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. Comput.*, 42(3):1245–1274, 2013.

24 Michael Freedman, László Lovász, and Alexander Schrijver. Reflection positivity, rank connectivity, and homomorphism of graphs. *J. Amer. Math. Soc.*, 20(1):37–51, 2007.

25 Leslie A. Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A complexity dichotomy for partition functions with mixed signs. *SIAM J. Comput.*, 39(7):3336–3402, 2010.

26 Heng Guo, Pinyan Lu, and Leslie G. Valiant. The complexity of symmetric Boolean parity Holant problems. *SIAM J. Comput.*, 42(1):324–356, 2013.

27 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Comb. Theory Ser. B*, 48(1):92–110, 1990.

28 Sangxia Huang and Pinyan Lu. A dichotomy for real weighted Holant problems. In *Proceedings of the 27th Conference on Computational Complexity*, pages 96–106, 2012.

29 Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

30 László Lovász. Operations with structures. *Acta Math. Hung.*, 18(3-4):321–328, 1967.

31 Leslie G. Valiant. Accidental algorithms. In *Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 509–517, 2006.

32 Leslie G. Valiant. Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008.

Polynomial-Time Rademacher Theorem, Porosity and Randomness

Alex Galicki

The University of Auckland, Auckland, New Zealand
agal629@aucklanduni.ac.nz

Abstract

The main result of this paper is a polynomial time version of Rademacher's theorem. We show that if $z \in \mathbb{R}^n$ is p-random, then every polynomial time computable Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at z. This is a generalization of the main result of [19].

To prove our main result, we introduce and study a new notion, p-porosity, and prove several results of independent interest. In particular, we characterize p-porosity in terms of polynomial time computable martingales and we show that p-randomness in $\mathbb{R}^n$ is invariant under polynomial time computable linear isometries.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Rademacher, porosity, p-randomness, differentiability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.30

1 Introduction

The topic of interactions between algorithmic randomness [11, 18] and computable analysis [17, 27] has been extensively studied in the recent years. The general idea is that classical theorems about properties holding almost everywhere in $\mathbb{R}^n$ have effective variants formulated in terms of algorithmic randomness. Differentiability of well-behaved functions is a sub-area that attracted particular interest of researchers (see [8, 13]). Most of results in this area are concerned with computable real functions of one variable. Less is known about functions of several variables (however, see [20, 15, 14]) and still less is known about differentiability and randomness in resource bounded settings [19].

The randomness notion this paper is concerned about is *p-randomness*, first studied by Wang [26]. It is usually defined in terms of polynomial time computable betting strategies.

In [19], Nies characterized p-randomness in terms of differentiability of polynomial time computable real-valued monotone functions of one variable. He showed that $z \in [0, 1]$ is p-random if and only if every polynomial time computable monotone function $f : [0, 1] \to \mathbb{R}$ is differentiable at z. Note that if $f : \mathbb{R} \to \mathbb{R}$ is a $K-$Lipschitz function, then $x \to f(x) + Kx$ is a monotone function. Hence the $\Rightarrow$ direction of this result also shows that polynomial time computable Lipschitz functions are differentiable at p-random reals.

The following classical result by Hans Rademacher [23] shows that Lipschitz real valued functions on $\mathbb{R}^n$ are almost everywhere differentiable.

▶ **Theorem 1** (Rademacher, 1919). *If $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz, then it is differentiable at almost every $x \in \mathbb{R}^n$ (with respect to the Lebesgue measure).*

In this paper we prove the following polynomial time version of Rademacher's theorem:

▶ **Theorem 2** (Polynomial time Rademacher). *If $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz and polynomial time computable, then it is differentiable at every p-random $x \in \mathbb{R}^n$.*

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 30; pp. 30:1–30:13

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

The notion of porosity, which originated in works of Denjoy, is crucial for this paper. Informally, x is a porosity point of $S \subseteq \mathbb{R}^n$ if it is possible to find relatively large balls disjoint from S (called *holes in* S) arbitrarily close to x. Most proofs of Rademacher's theorem, as well as our proof of the Theorem 1, have two distinct steps:

- The "one-dimensional" step, showing that directional derivatives (of a given Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$) exist almost everywhere. This part can be seen as concerned with differentiability of real functions of one variable.
- The step which shows that the set of points where the full derivative does not exist despite existence of some of the directional derivatives is negligible too.

Porosity can be observed and exploited in both steps. For real functions of one variable, porosity appears in sets where different types of derivatives disagree (see [9, 25] and [1]). In a polynomial time setting this phenomenon has been exploited by Nies in [19]. In the second step porosity appears in sets witnessing failures of linearity of directional derivatives (that is, when the directional derivative at a point as a function of direction is not a linear function). This particular phenomenon has been observed and studied for functions exhibiting Lipschitz-like regularity (for example, see [22] and [7]).

Since our main goal is to prove a polynomial-time version of Rademacher's theorem, we need a polynomial-time version of porosity. In the Section 3 we define a suitable notion, which we call *p-porosity*. It is worth mentioning that at least one effective version of porosity and its connections to algorithmic randomness has been studied before (see [6, 16]). We will briefly explain the difference between this notion of porosity and ours later in the paper.

The paper is structured as follows: in the Section 2 we review the relevant basic notions and define the notation used in the paper. In the Section 3 we define and study the notion of p-porosity. In the Section 4 we outline the proof of the Theorem 2.

2 Preliminaries and notation

In this paper we often have to go back and forth between the Cantor space and $\mathbb{R}^n$. Mostly, we use the standard notation (for notation related to the Cantor space and strings of finite length, please consult [18]). However, for the sake of readability and expressiveness, we will introduce some custom notation which is described below.

2.1 Dyadic cubes in $\mathbb{R}^n$, 1/3-shift trick

Let $\mathcal{D}^n$ denote the collection of half-open basic dyadic cubes in $\mathbb{R}^n$. That is

$$\mathcal{D}^n = \left\{2^{-k}([m_1, m_1+1) \times \cdots \times [m_n, m_n+1)) : \ k \in \mathbb{Z},\ m_1, \ldots, m_n \in \mathbb{Z}\right\}.$$

For $k \in \mathbb{Z}$, let $\mathcal{D}^n(k)$ denote the collection of basic dyadic cubes in $\mathbb{R}^n$ with its side length equal to 2^{-k}. If $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ is a linear isometry, let $\mathcal{D}_\Omega$ denote the set of images of elements of $\mathcal{D}^n$ under Ω. Analogously, let $\mathcal{D}_\Omega(k)$ be the collection of images of elements of $\mathcal{D}^n(k)$ under Ω. For $x \in \mathbb{R}^n$ and $i \in \mathbb{N}$, define $\mathcal{D}^n(x, i)$ (respectively, $\mathcal{D}_\Omega(x, i)$) to be the unique element of $\mathcal{D}^n(i)$ (respectively, $D_\Omega(i)$) containing x.

By $B(x, r)$ we denote the open ball in $\mathbb{R}^n$ with radius r and centered at x. The following proposition is known as the "$1/3-$shift trick" in $\mathbb{R}^n$.

▶ **Proposition 3** (cf. Theorem 3.8 in [24]). *For any ball $B = B(x, r) \subset \mathbb{R}^n$, there exists $k \in \mathbb{Z}$, $Q \in \mathcal{D}^n(k)$ and $t \in \{0, 1/3, 2/3\}^n$ such that $B \subset (Q + t)$ and $6r < 2^{-k} \leq 12r$.*

2.2 Binary expansion of elements in $\mathbb{R}^n$

Each real number $r \in [0,1)$ can be written in the form $r = \sum_{i\geq 0} r_i 2^{-i-1}$ where $r_i \in \{0,1\}$. We say $r_0 r_1 ...$ is the binary expansion of r. The binary expansion of r is unique unless r is a dyadic rational.

Let $x \in \mathbb{R}^n$. For every $1 \leq i \leq n$, let X_i denote the binary expansion of x_i, the $i-$th component of x. Then $X \in 2^\omega$ is the binary expansion of x if for all $1 \leq i \leq n$ and all j, $X(nj+i-1) = X_i(j)$.

Fix a positive $m \in \mathbb{N}$. Let $A \in 2^\omega$. We denote by $0._m A$ an element of $\mathbb{R}^m$, whose binary expansion is A. We omit the m subscript, when it is clear from the context.

Let $\sigma \in 2^{<\omega}$ with $|\sigma| = nk + m$ for some natural numbers n, k, m with $n > 0$ and $m < n$. Define $\{\sigma\}_n$ by $\{\sigma\}_n = \sigma\restriction_{nk}$. By $[\sigma]_n$ we denote the basic (open) dyadic cube (in $\mathbb{R}^n$) corresponding to $\{\sigma\}_n$.

2.3 Polynomial time computability and p-randomness

Intuitively, a function $f : \mathbb{R}^n \to \mathbb{R}$ is computable in polynomial time if for any $s \in \mathbb{N}$ and $x \in \mathbb{R}^n$, we can compute, uniformly in s and x, an approximate value e to $f(x)$ within an error 2^{-s} in time $O(s^k)$ for some constant k. For the rigorous definition, please see the Section 2.5 in [17]. This approach is equivalent to the one used in [19].

A *martingale* is a function $M\colon 2^{<\omega} \to \mathbb{R}_0^+$ such that $2M(\sigma) = M(\sigma 0) + M(\sigma 1)$ for all $\sigma \in 2^{<\omega}$. This notion of martingales is meant to formalize the intuitive notion of a betting strategy:

- $M(\sigma)$ represents the capital available after betting on bits of σ, which is always positive,
- betting the amount $\alpha \leq M(\sigma)$ on the next bit being 0, will result in losing α in the case when the next bit is 1 ($M(\sigma 1) = M(\sigma) - \alpha$) and winning α otherwise ($M(\sigma 0) = M(\sigma) + \alpha$).

We say that M *succeeds* on Z if $\limsup_n M(Z\restriction_n) = \infty$. For more details, please see [18].

▶ **Definition 4.** A martingale M is called *polynomial time computable* if from a string σ and $i \in \mathbb{N}$ we can in time polynomial in $|\sigma| + i$ compute an approximate value $(M(\sigma))_i$ to $M(\sigma)$ with $|M(\sigma) - (M(\sigma))_i| \leq 2^{-i}$.

▶ **Definition 5.** We say that $Z \in 2^\omega$ is *p-random* if no polynomial time martingale succeeds on Z. An element of $\mathbb{R}^n$ is said to be p-random if its binary expansion is p-random.

p-randomness is a polynomial time variant of *computable randomness* (see [18]). Computable randomness is usually defined in terms of succeeding of computable martingales. However, it is known that in the context of computable randomness, succeeding can be replaced with *divergence*. That is, $Z \in 2^\omega$ is not computably random iff there exists a computable martingale M with $\liminf_i M(Z\restriction_i) < \limsup_i M(Z\restriction_i)$. An analogous result holds for p-randomness and a somewhat stronger proposition will be shown later in this paper.

2.4 Martingales-measures correspondence and derivatives

We denote the Lebesgue measure (both on $\mathbb{R}^n$ and on 2^ω) by λ. By $\mathcal{M}_\lambda$ we denote the class of measures μ on $\mathbb{R}^n$ for which $\mu(A) \leq k \cdot \lambda(A)$ holds for some k and all Borel A.

Most of martingales considered in this paper are bounded. We will use repeatedly the following correspondence between measures from $\mathcal{M}_\lambda$ and bounded martingales.

▶ **Definition 6.** Let M be a martingale bounded from above. For all $\sigma \in 2^{<\omega}$ define

$$\mu_0([\sigma]_n) = M(\{\sigma\}_n)\lambda([\sigma]_n).$$

This defines a pre-measure on $[0,1]^n$. We can extend μ_0 to a measure $\mu \in \mathcal{M}_\lambda$ on $\mathbb{R}^n$ supported on a subset of the unit cube. We say μ is *a corresponding (to M) measure.*

For the other direction, let $\mu \in \mathcal{M}_\lambda$. We define *the corresponding (to μ) martingale M* by setting

$$M(\sigma) = \frac{\mu([\sigma]_n)}{\lambda([\sigma]_n)}.$$

Clearly, M is a bounded martingale.

▶ **Notation 7.** *Let μ be a measure on $\mathbb{R}^n$. Let $x \in \mathbb{R}^n$ and let $i \in \mathbb{N}$. Define*

$$\frac{\partial_2 \mu}{\partial_2 \lambda}(x,i) = \frac{\mu(\mathcal{D}^n(x,i))}{\lambda(\mathcal{D}^n(x,i))}$$

and

$$\frac{\partial_2 \mu}{\partial_2 \lambda}(x) = \lim_{i \to +\infty} \frac{\partial_2 \mu}{\partial_2 \lambda}(x,i).$$

If $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ is a linear isometry, we define

$$\frac{\partial_\Omega \mu}{\partial_\Omega \lambda}(x,i) = \frac{\mu(\mathcal{D}_\Omega(x,i))}{\lambda(\mathcal{D}_\Omega(x,i))}$$

and

$$\frac{\partial_\Omega \mu}{\partial_\Omega \lambda}(x) = \lim_{i \to +\infty} \frac{\partial_\Omega \mu}{\partial_\Omega \lambda}(x,i).$$

▶ **Notation 8.** *Let $n \geq 1$. By $e_1, \ldots, e_n$ we denote the unit vectors of the standard basis for $\mathbb{R}^n$.*

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function and let $v, x \in \mathbb{R}^n$. By $D_1 f(x), \ldots, D_n f(x)$ we denote the partial derivatives of f. We denote the directional derivative (in the direction of v) of f at x by $D_v f(x)$.

3 p-porosity

Let (X,d) be a metric space. $x \in X$ is said to be a *porosity point* of $S \subseteq X$ if

$$\mathbf{por}(x,S) = \limsup_{r \to 0} \gamma(x,r,S)/r > 0,$$

where $\gamma(x,r,S)$ is defined for any $r > 0$ as

$$\sup\{r' > 0 : \quad \text{for some } z \in X,\ B(z,r') \subseteq B(x,r) \text{ and } B(z,r') \cap S = \emptyset\}.$$

A set S is said to be *porous* if all its points are porosity points of S. A set is said to be *$\sigma-$porous* if it is a countable union of porous sets.

The following definitions are meant to formalize an efficient version of the above notion of porosity.

▶ **Definition 9.** Let C be a subset of 2^ω and let $Z \in C$. Define

$$\mathbf{por}_2(Z, C) = \liminf_{i\to\infty}\{|\sigma| - i \ : \ \sigma \succ Z\restriction_i \wedge \ [\sigma] \cap C = \emptyset\}.$$

If $\mathbf{por}_2(Z, C) < \infty$, then we say that Z is a *dyadic porosity point of* C.

Since we are interested in polynomial time computable betting strategies, we need to restrict our attention to subsets of 2^ω for which finding holes can be done in polynomial time.

▶ **Definition 10.** Let $A \subseteq 2^\omega$ be a Σ^0_1 set. We say A is *polynomial time computable* if there is a function $p : 2^{<\omega} \to \{0, 1\}$ computable in polynomial time such that $p(\sigma) = 1 \iff [\sigma] \subset A$ for all σ. Let $B \subseteq 2^\omega$ be a Π^0_1 set. We say it is polynomial time computable in if its complement is polynomial time computable.

▶ **Definition 11.** Let $X \in 2^\omega$. We say X is a *polynomial time porosity point (p-porosity point)* if there exists a polynomial time computable Π^0_1 set $A \subseteq 2^\omega$ such that X is a dyadic porosity point of A. If $C \subseteq A$ and $X \in C$, we say X is a p-porosity point of C.

We say $X \in 2^\omega$ is a *p-nonporosity point* if it is not a p-porosity point.

To show that $Z \in 2^\omega$ is a p-porosity point, it is sufficient to describe a polynomial-time algorithm for locating holes in some $S \subset 2^\omega$ arbitrarily close to Z. That is, to exhibit a function $p : 2^{<\omega} \to \{0, 1\}$ computable in polynomial time, such that

1. $p(\sigma) = 1 \iff [\sigma] \cap S = \emptyset$ and
2. Z is a dyadic porosity point of the complement of $\bigcup_{p(\sigma)=1}[\sigma]$.

▶ Remark. While our definitions admit straightforward generalizations to $\mathbb{R}^n$, for the sake of simplicity, we have defined our notion of p-porosity in terms of the Cantor space.

▶ Remark. As was mentioned in the introduction, one other effective version of porosity has been studied in the context of algorithmic randomness [6, 16]. $X \in 2^\omega$ is said to be a *porosity point* if there exists a Π^0_1 set $S \subseteq 2^\omega$ such that X is a dyadic porosity point of S. The main difference between this notion and p-porosity is that the latter requires a polynomial time algorithm for finding holes, while holes in a Π^0_1 set in general can only be enumerated.

3.1 p-porosity and polynomial time computable martingales

Since out main result, Theorem 2, is concerned with p-randomness, and we plan to use the notion of p-porosity extensively in the proof of it, we need to characterize the notion of p-porosity in terms of success sets of polynomial time computable martingales. This subsection is devoted to this task.

▶ Remark. A closely related notion, *p-genericity*, has been studied extensively (see [3], [2]). $Z \in 2^\omega$ is said to be *p-generic* if it does not belong to the boundary of any polynomial time computable Σ^0_1 subset of 2^ω. Since every p-porosity point belongs to the boundary of some polynomial time computable Σ^0_1 set, p-genericy implies p-nonporosity. Moreover, it is known that p-randomness implies p-genericity (see [4]). Hence, p-randomness implies p-nonporosity.

▶ **Definition 12.** Let μ be a measure on $\mathbb{R}^n$ and let $\epsilon > 0$. We say $x \in \mathbb{R}^n$ is *an $\epsilon-$oscillation point of μ* if for infinitely many $i \in \mathbb{N}$,

$$\left|\frac{\partial_2\mu}{\partial_2\lambda}(x, i) - \frac{\partial_2\mu}{\partial_2\lambda}(x, i + 1)\right| \geq \epsilon.$$

We say $x \in \mathbb{R}^n$ is *an oscillation point of μ* if x is an $\epsilon-$oscillation point of μ for some $\epsilon > 0$.

Analogously, we say $X \in 2^\omega$ is *an $\epsilon-$oscillation point of a martingale M* if for infinitely many $i \in \mathbb{N}$,

$$|M(X\restriction_i) - M(X\restriction_{i+1})| \geq \epsilon.$$

Let M be a martingale and let $\epsilon > 0$. By $\mathbf{Osc}(M, \epsilon)$ we denote the set of $\epsilon-$oscillation points of M. Finally, we let

$$\mathbf{Osc}(M) = \bigcup_{\epsilon > 0} \mathbf{Osc}(M, \epsilon).$$

▶ **Definition 13.** For $A, B \subseteq \mathbb{R}^n$ we say A and B are *$\epsilon-$separated by μ* if

$$\left| \frac{\mu(A)}{\lambda(A)} - \frac{\mu(B)}{\lambda(B)} \right| \geq \epsilon.$$

The following proposition provides a characterization of p-randomness in terms of $\epsilon-$oscillation points of polynomial time computable martingales.

▶ **Proposition 14.** *Let $Z \in 2^\omega$. The following are equivalent:*
1. *Z is p-random and*
2. *$Z \notin \boldsymbol{Osc}(M)$ for every bounded from above polynomial time computable martingale M.*

Proof Sketch. The (1) $\Rightarrow$ (2) direction is a polynomial time version of the Doob martingale convergence theorem. A straightforward adaptation of the proof of Theorem 7.1.3 from [10] suffices.

For the (2) $\Rightarrow$ (1) direction, suppose M is a polynomial time computable martingale succeeding on $Z \in 2^\omega$. We may assume M has the *saving property*, that is $M(\sigma\nu) \geq M(\sigma) - 1$ and $M(\sigma) > 1$ for all $\sigma, \nu \in 2^{<\omega}$. (See the proof of the Proposition 5.3.8 in [10])

Our proof is a suitable modification of the construction found in the proof of Theorem 4.2 from [13]. There, given a computable martingale M with the saving property, succeeding on Z, authors construct a computable martingale B that diverges on Z and for all $\sigma \in 2^{<\omega}$, $1 \leq B(\sigma) \leq 4$. It is easy to verify, that when M is polynomial time computable, B is polynomial time computable too. The construction turns the success of M into oscillations of B. It does so by having B alternating between two "phases": in the *up phase* B adds the capital that M risks, until $B(\sigma)$ reaches 3, while in the *down phase*, B subtracts the capital that M risks, until $B(\sigma)$ reaches 2. The required modification is following: the last bet of every up phase is a $1/4-$bet. It can be verified that for all $\sigma \in 2^{<\omega}$, $1 - 1/4 \leq B(\sigma) \leq 4 + 1/4$ and $Z \in \mathbf{Osc}(B, 1/4)$. ◀

▶ **Definition 15.** Let M be a martingale. We define $\mathcal{E}_{\geq}(M)$ to be the set of those X such that M does not make any losses while betting on X. More formally,

$$\mathcal{E}_{\geq}(M) = \{Z \ : \ \forall_i \ M(Z\restriction_{i+1}) \geq M(Z\restriction_i)\}.$$

The following proposition provides a characterization of p-porosity points in terms of martingales: p-porosity points are precisely those X for which there exists a martingale computable in polynomial time that succeeds on X without making any losses and places infinitely many ϵ-bets in the process.

▶ **Proposition 16.** *Let $Z \in 2^\omega$. The following two are equivalent:*
1. *Z is a p-porosity point, and*
2. *$Z \in \boldsymbol{Osc}(M) \cap \mathcal{E}_{\geq}(M)$ for some computable in polynomial time martingale M.*

Proof 1 ⇒ 2. Let A be a polynomial time computable Σ^0_1 set and let $p : 2^{<\omega} \to \{0,1\}$ be as in the Definition 10. Suppose Z is a dyadic porosity point of $2^\omega \setminus A$. Let $s = \mathbf{por}_2(Z, 2^\omega \setminus A)$. We define a martingale M in the following way. Let $M(\emptyset) = 1$ (by $\emptyset$ we denote the empty string). For all strings $\sigma \in 2^{<\omega}$ with $l = |\sigma| = k(s+1)$ for some $k > 1$, we let $M(\sigma) = M(\sigma\restriction_{l-1})$. Suppose $M(\sigma)$ has been defined, where $l = |\sigma| = k(s+1)$. If there is a string $\tau \succ \sigma$ of length $l+s$ such that $p(\tau) = 1$, then let $M(\tau) = 0$ and let $M(\sigma_1) = M(\sigma)\frac{2^s}{2^s-1}$ for all $\sigma_1 \succ \sigma$ with $\sigma_1 \neq \tau$ and $|\sigma_1| = s + l$. Otherwise, if such string τ does not exist, let $M(\sigma_1) = M(\sigma)$ for all $\sigma_1 \succ \sigma$ with $|\sigma_1| < (k+1)(s+1)$. M is clearly computable in polynomial time. Since Z is a dyadic porosity point of $2^\omega \setminus A$, for infinitely many i, we have

$$M(Z\restriction_{i+s}) - M(Z\restriction_i) \geq \frac{1}{2^s - 1}.$$

It follows that $Z \in \mathbf{Osc}\left(M, \frac{1}{s(2^s-1)}\right) \cap \mathcal{E}_{\geq}(M)$. ◀

Proof 2 ⇒ 1. Suppose $Z \in \mathbf{Osc}(M, \epsilon) \cap \mathcal{E}_{\geq}(M)$, where M is a computable in polynomial time martingale and $\epsilon > 0$. Let $s \in \mathbb{N}$ be such that $\epsilon > 2^{-s}$. For every $\sigma \in 2^{<\omega}$ with $\sigma \neq \emptyset$, let $\overline{\sigma}$ denote the string obtained from σ by flipping the last bit.

Define $p : 2^{<\omega} \to \{0,1\}$ by letting

- $p(\emptyset) = 0$ and
- for all $\sigma \neq \emptyset$, if $\left(M(\sigma) - M(\sigma\restriction_{|\sigma|-1})\right)_s > 0$, then $p(\overline{\sigma}) = 1$. Otherwise, let $p(\overline{\sigma}) = 0$.

Since $\left(M(\sigma) - M(\sigma\restriction_{|\sigma|-1})\right)_s > 0$ is decidable in polynomial time, p is computable in polynomial time too. Hence the set $A = 2^\omega \setminus \bigcup_{p(\sigma)=1}[\sigma]$ is polynomial time computable. For infinitely many i, we have $M(Z\restriction_{i+1}) - M(Z\restriction_i) > 2^{-s}$ and hence $p\left(\overline{Z\restriction_{i+1}}\right) = 1$. It follows that Z is a dyadic porosity point of A. ◀

4 Polynomial-time Rademacher's theorem

4.1 Overview of the proof

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a Lipschitz function. Let us denote by $N(f)$ the set of points where f is not differentiable. Classical Rademacher's theorem asserts that it is a Lebesgue nullset. This can be proven in two steps.

1. Firstly, fix a countable set of unit vectors $V \subset \mathbb{R}^n$. Let $N(V, f) \subset \mathbb{R}^n$ denote the set where $D_v f(x)$ does not exist for at least one $v \in V$. A.e. differentiability of real-valued Lipschitz functions of one variable in conjunction with Fubini's theorem implies that this set is a Lebesgue nullset.
2. Secondly, consider the set $N(f) \setminus N(V, f)$. It can be proven that this set is $\sigma-$porous provided V is not empty (for example, see Theorem 3.1 in [5] and Theorem 2 in [21]). This concludes the proof, since $\sigma-$porous sets are Lebesgue nullsets.

Our proof of Theorem 2 follows a similar path. Firstly, let V_p be the set of polynomial time computable unit vectors in $\mathbb{R}^n$. Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a polynomial time computable Lipschitz function. We need to show that $N(f)$ contains no p-random elements. Just like in the classical case outlined above, we show this by splitting $N(f)$ in two parts - $N(f) \setminus N(V_p, f)$ and $N(V_p, f)$ - and then showing that neither of them contains a p-random element.

The proof has three nontrivial and relatively self-contained parts:

- Firstly, we show a result of independent interest. In the subsection 4.2 we prove that p-randomness in $\mathbb{R}^n$ is invariant under linear isometries computable in polynomial time.

It is worth mentioning that the one-dimensional version of this result follows from results in [12]. Higher dimensional result in this paper, requires, however, quite a different approach.

- Then, we show that p-randomness of z is sufficient for existence of partial derivatives of f at z. This is an adaptation of the one-dimensional proof from [19]. Existence of directional derivatives $D_v f(z)$ where $v \in V_p$ follows the preservation property mentioned in the previous point. This concludes the proof that $N(V_p, f)$ contains no p-random elements.
- Finally, we demonstrate that $N(f) \setminus N(V_p, f)$ contain no p-random points. This is accomplished by a careful analysis of structural properties of $N(f) \setminus N(V_p, f)$ and showing that binary expansions of elements of this set are p-porosity points.

Due to size limitations, this paper contains the full proof of the first part only (bar the proof of the technical lemma from the Section 4.2.2).

4.2 Invariance of p-randomness under linear isometries computable in polynomial time

In this subsection we will use the following notational convention.

▶ **Notation 17.** *Let μ be a measure on $\mathbb{R}^n$, and let $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ be a linear isometry. By μ_Ω we denote the measure defined by $\mu_\Omega(A) = \mu(\Omega(A))$ for all Borel A.*

If M is a martingale corresponding to μ, by M_Ω we denote the martingale corresponding to μ_Ω.

The main result of this subsection is that p-randomness is invariant under polynomial time computable linear isometries. Let us examine how an analogous result can be shown for computable randomness. Suppose $z \in \mathbb{R}^n$ be not computably random and let $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ be a computable linear isometry. We want to show that $\Omega^{-1}(z)$ is not computably random. Let M be a bounded computable martingale diverging on Z (where $z = 0.Z$) and let μ be a corresponding measure on $\mathbb{R}^n$. Define $y = \Omega^{-1}(z)$ and let Y be the binary expansion of y. Observe that M_Ω is also a bounded computable martingale. There are two possibilities:
(A) Either M_Ω diverges on Y, in which case Y is not computably random, or
(B) M_Ω converges on Y and then y belongs to the set

$$A = \left\{ x : \ \frac{\partial_2 \mu_\Omega}{\partial_2 \lambda}(x) \text{ exists and } \frac{\partial_\Omega \mu_\Omega}{\partial_\Omega \lambda}(x) \text{ does not exist} \right\}.$$

In this case it is possible to show that y is a porosity point of some subset of A and use this information to conclude that y is not computably random.

A similar argument can be made about p-randomness. However, there are two additional obstacles. Firstly, it is not clear what (additional) conditions on M and Ω ensure that M_Ω is polynomial time computable. Secondly, the porosity mentioned in (B) would have to be replaced with p-porosity. A significant portion of this section is dedicated to address those two problems. The plan is following:

- In the Subsection 4.2.1 we show that $\epsilon-$oscillation points of M_Ω are not p-random, even if it is not known whether M_Ω is computable in polynomial time;
- In the Subsection 4.2.2 we prove a technical lemma related to linear transformations and $\epsilon-$oscillation;
- Finally, in the Subsection 4.2.3 we combine those ideas to prove our main invariance theorem.

4.2.1 Betting on ϵ-oscillation points of M_Ω

▶ **Lemma 18.** *Let $A, B \subseteq \mathbb{R}^n$ be Borel with $A \subseteq B$ and $\lambda(B) > 0$. Let μ be a measure on $\mathbb{R}^n$ such that for some $k \in \mathbb{N}$, $\mu(C) \leq k\lambda(C)$ for all C. Suppose $\frac{\lambda(B\setminus A)}{\lambda(A)} \leq \epsilon$ for some $\epsilon \in \mathbb{R}$. Then*

$$\left|\frac{\mu(B)}{\lambda(B)} - \frac{\mu(A)}{\lambda(A)}\right| \leq 2k \cdot \epsilon.$$

Proof.

$$\begin{aligned}\left|\frac{\mu(B)}{\lambda(B)} - \frac{\mu(A)}{\lambda(A)}\right| &= \left|\frac{\mu(B)\lambda(A) - \mu(B)\lambda(B) + \mu(B\setminus A)\lambda(B)}{\lambda(B)\,\lambda(A)}\right| \\ &= \left|\frac{\mu(B)}{\lambda(B)}\frac{\lambda(B)-\lambda(A)}{\lambda(A)} + \frac{\mu(B\setminus A)}{\lambda(A)}\right| \\ &\leq 2k \cdot \epsilon.\end{aligned}$$

◀

▶ **Lemma 19** (Approximation lemma)**.** *Let M be a computable in polynomial time martingale bounded above by some $k \in \mathbb{N}$. Let μ be a corresponding measure on $\mathbb{R}^n$. Let $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ be a computable in polynomial time linear isometry. Fix $s \in \mathbb{N}$. There exists function $M_{\Omega,s} : 2^{<\omega} \to \mathbb{R}$ computable in polynomial time such that for all σ*

$$|M_{\Omega,s}(\sigma) - M_\Omega(\sigma)| \leq 2^{-s}.$$

Proof. This is a consequence of Lemma 18. For a given σ, we can find in polynomial time a finite collection of dyadic basic cubes $(D(\sigma)_i)_{i\in\mathbb{N}}$ such that

$$\frac{\lambda(\Omega([\sigma]_n) \setminus \bigcup_i D(\sigma)_i)}{\lambda(\bigcup_i D(\sigma)_i)} \leq \frac{2^{-s-1}}{k},$$

so that

$$\left|\frac{\mu(\Omega([\sigma]_n))}{\lambda(\Omega([\sigma]_n))} - \frac{\mu(\bigcup_i D(\sigma)_i)}{\lambda(\bigcup_i D(\sigma)_i)}\right| \leq 2^{-s}.$$

Define $M_{\Omega,s}(\sigma) = \frac{\mu(\bigcup_i D(\sigma)_i)}{\lambda(\bigcup_i D(\sigma)_i)}$. This function is computable in polynomial time and the following holds for every σ:

$$|M_\Omega(\sigma) - M_{\Omega,s}(\sigma)| = \left|\frac{\mu(\Omega([\sigma]_n))}{\lambda(\Omega([\sigma]_n))} - \frac{\mu(\bigcup_i D(\sigma)_i)}{\lambda(\bigcup_i D(\sigma)_i)}\right| \leq 2^{-s}.$$

◀

▶ **Lemma 20.** *Let M be a polynomial time computable martingale bounded above by some $k \in \mathbb{N}$ and let $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ be a polynomial time computable linear isometry. For every $\epsilon > 0$, there exists a polynomial time computable martingale H_ϵ such that every $\epsilon-$oscillation point of M_Ω is an $\epsilon/2-$oscillation point of H_ϵ.*

Proof. Let s be such that $2^{-s+1} < \epsilon$. By Lemma 19, there exists a polynomial time computable function $M_{\Omega,s}$ such that for all σ

$$|M_{\Omega,s}(\sigma) - M_\Omega(\sigma)| \leq 2^{-s}.$$

We define H_ϵ as following. We let $H_\epsilon(\emptyset) = M_{\Omega,s}(\emptyset)$. For any σ, suppose $H_\epsilon(\sigma)$ has been defined. We define $\alpha(\sigma) = H_\epsilon(\sigma) - \frac{1}{2}(M_{\Omega,s}(\sigma 0) + M_{\Omega,s}(\sigma 1))$ and we let

$$H_\epsilon(\sigma 0) = M_{\Omega,s}(\sigma 0) + \alpha(\sigma), \text{ and } H_\epsilon(\sigma 1) = M_{\Omega,s}(\sigma 1) + \alpha(\sigma).$$

It is easy to verify that H_ϵ is a polynomial time computable martingale. Now suppose $|M_\Omega(\sigma) - M_\Omega(\sigma 1)| \geq \epsilon$. We have

$$|H_\epsilon(\sigma) - H_\epsilon(\sigma 1)| = \frac{1}{2}\,|M_{\Omega,s}(\sigma 1) - M_{\Omega,s}(\sigma 0)| \geq \frac{1}{2}(2\epsilon - 2^{-s+1}) > \epsilon/2.$$

The case when $|M_\Omega(\sigma) - M_\Omega(\sigma 0)| \geq \epsilon$ is handled analogously. ◀

4.2.2 A technical lemma

Let M be a martingale computable in polynomial time, bounded from above, and let μ be a corresponding measure on $\mathbb{R}^n$. Suppose $y \in \mathbb{R}^n$ is an $\epsilon-$oscillation point of μ. It can be easily shown that for any $k > 0$ and for infinitely many i, $\mathcal{D}^n(y, i)$ contains two dyadic cubes from $\mathcal{D}^n(i + k)$, that are $\epsilon-$ separated by μ.

Now consider a linear isometry $\Omega : \mathbb{R}^n \to \mathbb{R}^n$. Suppose that for some $k > 0$, $\epsilon > 0$ and for infinitely many i, $D = \mathcal{D}^n(y, i)$ contains two cubes $D_1, D_2 \in \mathcal{D}_\Omega(i + k)$, that are $\epsilon-$separated by μ. In general, this does not imply that y is an oscillation point of μ. However, the following technical lemma shows that if y is not an oscillation point of μ, then it is a p-porosity point.

▶ **Lemma 21.** *Let M be a martingale computable in polynomial time, bounded from above. Let μ be a corresponding measure on $\mathbb{R}^n$. Let $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ be a linear isometry. Let $y \in \mathbb{R}^n$ with $Y \in 2^\omega$ being its binary expansion. Suppose that for some $k > 0$, $\epsilon > 0$ and for infinitely many i, $D = \mathcal{D}^n(y, i)$ contains two cubes $D_1, D_2 \in \mathcal{D}_\Omega(i + k)$, that are $\epsilon-$separated by μ. If y is not an oscillation point of μ, then Y is a p-porosity point.*

4.2.3 Invariance theorems

▶ **Theorem 22.** *Let $z \in [0, 1]^n$ and let $r \in \mathbb{R}$ be a polynomial time computable real. Suppose z is not p-random. Then $z + re_i$ is not p-random for any $1 \leq i \leq n$.*

Proof. Since we are only interested in the question whether $z + re_i$ is p-random or not, we may assume that z and r are such that $z + re_i \in [0, 1]^n$.

Without loss of generality we may assume that every component of z is p-random and $n > 1$ (otherwise, the required result follows from preservation properties proven in [12]). Fix $i \in \mathbb{N}$ with $1 \leq i \leq n$ and define $\Phi, \Omega : \mathbb{R}^n \to \mathbb{R}^n$ by $\Omega(x) = x - re_i$ and $\Phi = \Omega^{-1}$. Let $y = \Phi(z)$. Let Z be the binary expansion of z and let Y be the binary expansion of y. Let M be the polynomial time computable martingale M from the (sketch of the)proof of the Proposition 14 such that Z is an $\epsilon-$oscillation point of M for some $\epsilon > 0$. M makes an infinite number of ϵ bets along Z. However, we need a modified version of M (which we also call M). First, let us introduce some notation. Let μ denote the measure corresponding to M. For every $\sigma \in 2^{<\omega}$ with $[\sigma]_n \in \mathcal{D}(j)$, let $D^1(\sigma), D^2(\sigma) \in \mathcal{D}_\Omega(j)$ be such that $[\sigma]_n \subseteq D^1(\sigma) \cup D^2(\sigma)$. $D^1(\sigma)$ and $D^2(\sigma)$ are not necessarily unique, but that does not matter. Our martingale M, instead of making an ϵ bet, waits until its input σ is such that $\frac{\lambda([\sigma]_n \cap D^1(\sigma))}{\lambda([\sigma]_n)} \geq 1/3$ and $\frac{\lambda([\sigma]_n \cap D^2(\sigma))}{\lambda([\sigma]_n)} \geq 1/3$ (this will occur sooner or later since all components of z are p-random). Once such input σ is found (with $[\sigma]_n \in \mathcal{D}(j)$ for some j), M places two $\epsilon/2$ bets on $\tau_1, \tau_2 \succ \sigma$ such that $[\tau_1]_n, [\tau_2]_n \in \mathcal{D}(j + 2)$, $[\tau_1]_n \subseteq [\sigma]_n \cap D^1(\sigma)$ and $[\tau_2]_n \subseteq [\sigma]_n \cap D^2(\sigma)$.

What is important in this construction is that for infinitely many i, both $D^1(Z{\restriction_i})$ and $D^2(Z{\restriction_i})$ contain two elements of $\mathcal{D}(i + 2)$ that are $\epsilon/2-$separated by μ and either $y \in \Phi(D^1(Z{\restriction_i}))$ or $y \in \Phi(D^2(Z{\restriction_i}))$.

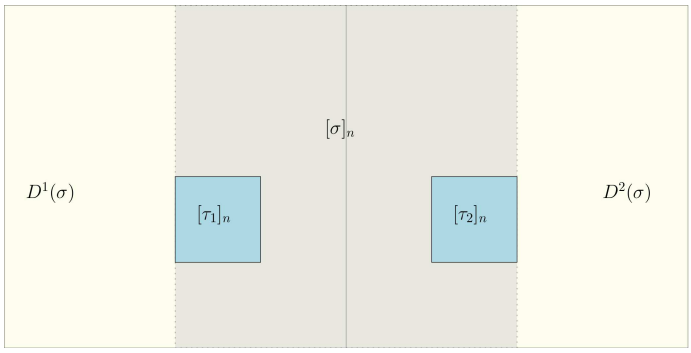


Figure 1 A particular betting pattern employed in the proof of the Theorem 22.

Clearly M is computable in polynomial time and Z is an $\epsilon/2-$oscillation point of M.

Consider the martingale M_Ω. By the Lemma 21, either $Y \in \mathbf{Osc}(M_\Omega)$ or Y is a p-porosity point. In both cases Y is not p-random. ◀

▶ Remark. There is an important implication of the above theorem. In those cases where we are only concerned whether some $x \in \mathbb{R}^n$ is p-random or not, we can always use the 1/3-shift trick freely. That is, since for every i, $x + 1/3e_i$ is p-random iff x is p-random, instead of x we can always consider a suitable shift of x. This will be used below, in the proof of our main result of this subsection.

▶ **Theorem 23.** *Let $\Omega : \mathbb{R}^n \to \mathbb{R}^n$ be a polynomial time computable linear isometry. Let $z \in [0,1]^n$. z is p-random iff $\Omega(z)$ is p-random.*

Proof. Since Ω^{-1} is polynomial time computable linear isometry as well, it is only required to show that if z is not p-random, then $\Omega^{-1}(z)$ is not p-random either. Again, we may assume $\Omega^{-1}(z) \in [0,1]^n$.

Let Z be the binary expansion of z. Let M be a bounded polynomial time computable martingale such that Z is an $\epsilon-$oscillation point of M for some $\epsilon > 0$. Define $\Phi = \Omega^{-1}$, let $y = \Phi(z)$ and let Y be the binary expansion of y.

Consider the martingale M_Ω and its corresponding measure μ_Ω. If y is an oscillation point of μ_Ω, Y is not p-random. Suppose y is not an oscillation point of μ_Ω.

There are infinitely many j, such that $\mathcal{D}_\Phi(j, y)$ and $\mathcal{D}_\Phi(j+1, y)$ are $\epsilon-$separated by μ_Ω. By the 1/3-shift trick and by Theorem 22, we may assume that for infinitely many such j's, $\mathcal{D}_\Phi(j, y)$ is contained in $\mathcal{D}^n(y, j - \hat{p})$ for some fixed $\hat{p}$. In that case, by the Lemma 21, Y is a p-porosity point and hence not p-random. ◀

4.3 Existence of directional derivatives

To prove our main result about directional derivatives, we need the following proposition:

▶ **Proposition 24.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a polynomial time computable Lipschitz function. If $z \in \mathbb{R}^n$ is p-random, then $D_n f(z)$ exists.*

▶ Remark. The proof of the above proposition is a generalization of the proof of the $\Rightarrow$ direction of the Theorem 4 in [19]. The most technically challenging part required by the proof is the $\Leftarrow$ part of van Lambalgen's theorem for p-randomness. That is, we had to show that if there is an oracle martingale computable in polynomial time diverging on A while having an oracle access to B, then there is a martingale computable in polynomial time succeeding on $A \oplus_n B$.

▶ **Theorem 25.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a Lipschitz function computable in polynomial time. Let $u \in V_p$ and let $x \in \mathbb{R}^n$ be p-random.*

The directional derivative $D_u f(x)$ exists.

Proof. Let $\Theta : \mathbb{R}^n \to \mathbb{R}^n$ be a change of basis map, such that $\Theta(e_1) = u$. We may assume it is computable in polynomial time. Define $z = \Theta^{-1}(x)$. By the Theorem 23, z is p-random too.

Define $g = f \circ \Theta$. g is a Lipschitz function computable in polynomial time. Then $D_u f(x) = D_1 g(z)$ and we know that $D_1 g(z)$ exists. ◀

4.4 Linearity of directional derivatives

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a Lipschitz function computable in polynomial time. Recall that $N(f) \setminus N(V_p, f)$ is the set of points $x \in \mathbb{R}^n$ such that $D_v f(x)$ exists for all unit vectors v computable in polynomial time but f is not differentiable at x.

Let $u, v \in \mathbb{R}^n$. Define $D(f, u, v) \subseteq \mathbb{R}^n$ as the set of such x that $D_u f(x), D_v f(x)$ and $D_{u+v} f(x)$ exist, but

$$D_u f(x) + D_v f(x) \neq D_{u+v} f(x).$$

Since f is Lipschitz and V_p is dense in the set of unit vectors, it is known that

$$N(f) \setminus N(V_p, f) = \bigcup_{u,v \in V_p} D(f, u, v).$$

The following proposition is the last bit required to prove the Theorem 2:

▶ **Proposition 26.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a Lipschitz function computable in polynomial time. Let $x \in \mathbb{R}^n$ with $X \in 2^\omega$ being its binary expansion. If $x \in N(f) \setminus N(V_p, f)$, then X is a p-porosity point.*

References

1. G. Petruska A. M. Bruckner, M. Laczkovich and B. S. Thomson. Porosity and approximate derivatives. *Canadian Journal of Mathematics*, 38:1149–1180, 1986. doi:10.4153/CJM-1986-058-7.
2. K. Ambos-Spies, H. Fleischhack, and H. Huwig. *P-generic sets*, pages 58–68. Springer Berlin Heidelberg, 1984. doi:10.1007/3-540-13345-3_5.
3. K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial time computable sets. *Theoretical Computer Science*, 51(1):177–204, 1987. doi:10.1016/0304-3975(87)90053-3.
4. K. Ambos-Spies, S. A. Terwijn, and Z. Xizhong. *Resource bounded randomness and weakly complete problems*, pages 369–377. Springer Berlin Heidelberg, 1994. doi:10.1007/3-540-58325-4_201.
5. D. N. Bessis and F. H. Clarke. Partial subdifferentials, derivatives and Rademacher's theorem. *Transactions of AMS*, 351(7):2899–2926, 1999.
6. Miller J. Bienvenu L., Hölzl R. and Nies A. Denjoy, Demuth and density. *Journal of Mathematical Logic*, 14(01):1450004, 2014. doi:10.1142/S0219061314500044.
7. J. M. Borwein and X. Wang. Cone-monotone functions: Differentiability and continuity. *Canadian Journal of Mathematics*, 57:961–982, 2005. doi:10.4153/CJM-2005-037-5.
8. V. Brattka, J. Miller, and A. Nies. Randomness and differentiability. *Transactions of the AMS*, 368:581–605, 2016. arXiv version at arxiv.org/abs/1104.4465.

9 A. M. Bruckner and B. S. Thomson. Porosity estimates for the Dini derivatives. *Real Anal. Exch.*, 9:508–538, 1984.

10 R. Downey and D. Hirschfeldt. *Algorithmic randomness and complexity*. Springer-Verlag, Berlin, 2010. 855 pages.

11 R. G. Downey and D. R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Springer-Verlag, 2010.

12 Stephen A. Fenner. Functions that preserve p-randomness. *Inf. Comput.*, 231:125–142, October 2013. doi:10.1016/j.ic.2013.08.009.

13 C. Freer, B. Kjos-Hanssen, A. Nies, and F. Stephan. Algorithmic aspects of Lipschitz functions. *Computability*, 3(1):45–61, 2014. doi:10.3233/COM-14025.

14 A. Galicki. *Randomness and Differentiability of Convex Functions*, pages 196–205. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-20028-6_20.

15 A. Galicki. Effective Brenier Theorem: Applications to Computable Analysis and Algorithmic Randomness. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS'16, pages 720–729, New York, NY, USA, 2016. ACM. doi:10.1145/2933575.2933596.

16 M. Khan. Lebesgue density and $\prod^0_1$ classes. *Journal of Symbolic Logic*, 81(1):80–95, 2016.

17 Ker-I Ko. *Complexity theory of real functions*. Birkhauser Boston Inc., 1991.

18 A. Nies. *Computability and randomness*, volume 51 of *Oxford Logic Guides*. Oxford University Press, Oxford, 2009. doi:10.1093/acprof:oso/9780199230761.001.0001.

19 André Nies. Differentiability of polynomial time computable functions. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 602–613, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2014.602.

20 N. Pathak, C. Rojas, and S. G. Simpson. Schnorr randomness and the Lebesgue differentiation theorem. *Proc. Amer. Math. Soc.*, 142(1):335–349, 2014. doi:10.1090/S0002-9939-2013-11710-7.

21 D. Preiss and L. Zajíček. Directional derivatives of lipschitz functions. *Israel Journal of Mathematics*, 125(1):1–27, 2001. doi:10.1007/BF02773371.

22 J. Lindenstrauss, D. Preiss and J. Tišer. *Fréchet Differentiability of Lipschitz Functions and Porous Sets in Banach Spaces*. Annals of Mathematics Studies. Princeton University Press, 2012.

23 H. Rademacher. Über partielle und totale Differenzierbarkeit von Funktionen mehrerer Variabeln und über die Transformation der Doppelintegrale. *Math. Ann.*, 79(1):340–359, 1919.

24 O. Tapiola. Random and non-random dyadic systems in doubling metric spaces, 2012. MSc thesis. URL: http://hdl.handle.net/10138/37603.

25 Brian S. Thomson. Real functions. Lecture Notes in Mathematics. 1170. Berlin etc.: Springer-Verlag. VII, 229 p. DM 31.50 (1985)., 1985. doi:10.1007/BFb0074380.

26 Y. Wang. *Randomness and Complexity*. PhD dissertation, University of Heidelberg, 1996.

27 K. Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

A QPTAS for the General Scheduling Problem with Identical Release Dates*

Antonios Antoniadis[1], Ruben Hoeksma[2], Julie Meißner[3], José Verschae[4], and Andreas Wiese[5]

1 **Department of Computer Science, University of Bonn, Bonn, Germany**
antoniad@cs.uni-bonn.de
2 **Center for Mathematical Modeling, Universidad de Chile, Santiago, Chile**
rhoeksma@dim.uchile.cl
3 **Institut für Mathematik, Technische Universität Berlin, Berlin, Germany**
jmeiss@math.tu-berlin.de
4 **Facultad de Matemáticas & Escuela de Ingeniería, Pontificia Universidad Católica de Chile, Santiago, Chile**
jverschae@uc.cl
5 **Department of Industrial Engineering & Center for Mathematical Modeling, Universidad de Chile, Santiago, Chile**
awiese@dii.uchile.cl

Abstract

The General Scheduling Problem (GSP) generalizes scheduling problems with sum of cost objectives such as weighted flow time and weighted tardiness. Given a set of jobs with processing times, release dates, and job dependent cost functions, we seek to find a minimum cost preemptive schedule on a single machine. The best known algorithm for this problem and also for weighted flow time/tardiness is an $O(\log \log P)$-approximation (where P denotes the range of the job processing times), while the best lower bound shows only strong NP-hardness. When release dates are identical there is also a gap: the problem remains strongly NP-hard and the best known approximation algorithm has a ratio of $e + \epsilon$ (running in quasi-polynomial time). We reduce the latter gap by giving a QPTAS if the numbers in the input are quasi-polynomially bounded, ruling out the existence of an APX-hardness proof unless $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{\mathrm{polylog(n)}})$. Our techniques are based on the QPTAS known for the UFP-Cover problem, a particular case of GSP where we must pick a subset of intervals (jobs) on the real line with associated heights and costs. If an interval is selected, its height will help cover a given demand on any point contained within the interval. We reduce our problem to a *generalization* of UFP-Cover and use a sophisticated divide-and-conquer procedure with interdependent non-symmetric subproblems.

We also present a pseudo-polynomial time approximation scheme for two variants of UFP-Cover. For the case of agreeable intervals we give an algorithm based on a new dynamic programming approach which might be useful for other problems of this type. The second one is a resource augmentation setting where we are allowed to slightly enlarge each interval.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Generalized Scheduling, QPTAS, Unsplittable Flows

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.31

* This work was partially funded by Nucleo Milenio Información y Coordinación en Redes ICM/FIC RC130003, Conicyt PII Nr 20150140, and Fondecyt Nr 11140579.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 31; pp. 31:1–31:14

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

1 Introduction

The *General Scheduling Problem* (GSP) considers scheduling jobs with job dependent cost functions in a very general setting. We are given a single machine and a set of jobs J, where each job j has a release date $\rho_j \in \mathbb{N}$, a processing time $p_j \in \mathbb{N}$, and a cost function $f_j : \mathbb{N} \to \mathbb{N}_0 \cup \{\infty\}$ that is non-decreasing. The goal is to find a preemptive schedule on the machine that minimizes the total cost $\sum_{j \in J} f_j(C_j)$, where C_j is the completion time of job j in the computed schedule.

With arbitrary cost functions for the jobs, we have a lot of modeling power, which we believe makes the problem worth studying. In fact, we can model many scheduling objectives that were also studied separately, such as weighted flow time (each job j has weight w_j and $f_j(C_j) = w_j(C_j - \rho_j)$) or weighted tardiness (each job j additionally has a deadline d_j and $f_j(C_j) = \max\{w_j(C_j - d_j), 0\}$). The best known result for GSP is a $O(\log \log P)$-approximation [5] (where P denotes the range of the processing time) and no better polynomial time results are known for any of the mentioned special cases. The best known lower bound shows only strong NP-hardness [13] (even in the case without release dates), thus leaving a large gap compared to the $O(\log \log P)$-approximation. Even if all jobs have identical release dates there is a gap in our understanding: the best known results are a $(4 + \epsilon)$-approximation in polynomial time [11] and an $(e + \epsilon)$-approximation in quasi-polynomial time [12]. It is open whether this case is APX-hard. In this paper we settle the latter question: for GSP with identical release dates we present a QPTAS, i.e., a $(1 + \epsilon)$-approximation algorithm with a running time of $n^{\log(n)^{O(1)}}$ for any constant $\epsilon > 0$. This implies that the problem is *not* APX-hard, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{\mathrm{poly}(\log n)})$.

In this extended abstract, many proofs and details had to be omitted due to space constraints.

1.1 General Scheduling Problem and UFP-Cover

For identical release dates, GSP is purely a sequencing problem, since a solution cannot profit from preempting jobs or leaving idle-time. Assuming that $\rho_j = 0$ for each j, the whole schedule finishes at time $T := \sum_j p_j$. Using the viewpoint from [5], we can see this problem as a covering problem. In any feasible solution, for each time t, we need that the total processing time of jobs finishing after time t is at least $D_t := T - t$. We can think of D_t as the *demand* of time point t. Now, we rephrase the problem as follows. For each job j select a completion time C_j such that for each t' the total processing time of the jobs unfinished at time t' is at least $D_{t'}$. We say that job j is *unfinished* or *active* during the interval $[0, C_j)$. An easy proof shows that, for each such choice of completion times, there exists a schedule in which every job j is finished by its completion time C_j [5].

An important special case arises when the cost function f_j of each job j attains only one of three values: zero in an interval $[0, r_j)$ (r_j should not to be confused with the release date $\rho_j = 0$), a job dependent value c_j in an interval $[r_j, d_j)$, and ∞ in $[d_j, \infty)$. In this setting, we can assume that the optimal solution selects either $[0, r_j)$ or $[0, d_j)$ to be the interval during which j is active. Moreover, we can simply remove p_j from the demand at each time $[0, r_j)$, which leaves as the only decision for j whether we pay c_j and cover p_j units of demand during $[r_j, d_j)$, or not. Thus, this special case can be reduced to the *Unsplittable Flow on a Path (UFP)-Cover* problem. In UFP-Cover, we are given a set of jobs J, each job described by a cost c_j, a size p_j, and the interval $[r_j, d_j)$ and, for each time point t, a demand D_t. The goal is to select a subset J' of the jobs such that, for each time point t, the total size of the jobs $j \in J'$ with $t \in [r_j, d_j)$ is at least D_t. Note that we do not

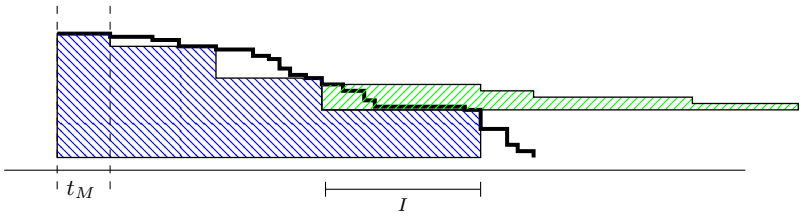


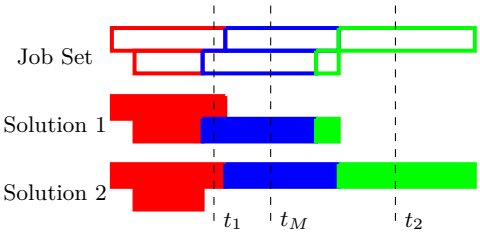
Figure 1 The bold curve denotes the size profile of job parts selected by the optimal solution that cross t_M. The blue step function shows an underestimating profile that approximates the former curve. The height of the green area (subprofile) is an (under-)estimation of the size of job parts in the optimal solution for all jobs that have a part covering t_M and whose right end point lies at interval I (i.e., the fourth step of the blue function).

require that the demand function D_t is non-increasing (but by adding jobs of zero cost one could assume this w.l.o.g.). The best known results for UFP-Cover are a 4-approximation in polynomial time [6, 8] and a QPTAS which requires the input data to be quasi-polynomially bounded [12]. Since there is the QPTAS, it is natural to conjecture that also a PTAS exists. In this paper, we make progress towards this by presenting pseudo-polynomial time approximation schemes for the settings of agreeable intervals, i.e., when for any two jobs j, j' we have that $r_j \leq r_{j'} \Rightarrow d_j \leq d_{j'}$, and for a resource augmentation setting, where we are allowed to increase each given interval $[r_j, d_j)$ by a factor of $1 + \mu$ for an arbitrarily small $\mu > 0$ while the compared optimal solution cannot do this.

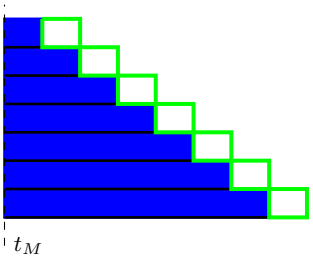
1.2 Our Contribution

Our first result is a QPTAS for GSP with identical release dates, assuming that all numbers in the input are quasi-polynomially bounded. We reduce GSP to a generalization of UFP-Cover. This *generalized UFP-Cover* problem is defined like regular UFP-Cover, but now each job j consists of K *parts*. More precisely, for each job j we are given an integral starting time r_j, a size p_j, up to K many integral end times $r_j < d_j^1 < d_j^2 < ... < d_j^K$, and corresponding accumulated costs $c_j^1, c_j^2, ..., c_j^K$. Jobs can be selected or not. If a job is not selected its cost is zero and it does not contribute to cover any demand. If job j is selected, we can choose to *extend it up to any part* $i \in \{1, \dots, K\}$, which means that then it is active during $[r_j, d_j^i)$. In this case we pay c_j^i for this job while it contributes to cover p_j units of demand to each time within $[r_j, d_j^i)$. The objective is to cover all demand D_t while minimizing the total cost. Notice that if $K = 1$ then we recover the UFP-Cover problem. On the other hand, we show that by losing a factor of $1 + \epsilon$ in the objective we can assume that $K = 1/\epsilon^2$.

Starting with this, we extend the known QPTAS for UFP-Cover, which works as follows. We consider the jobs crossing the middle time point t_M; denote them by J_M. They are split into $(\log n)^{O_\epsilon(1)}$ groups according to size and cost. For each group and each time t, we consider the total size of the jobs in the group crossing time point t in the optimal solution. This yields a function that is increasing from time 0 to time t_M, and decreasing from t_M to T. This function can be underestimated by a step-function (*profile*) with $O(1/\delta)$ (where $\delta = O_\epsilon(1)$) many steps (see the blue curve in Figure 1). One first guesses the step-function and then selects jobs that cover the demand given by this step-function greedily (which is essentially optimal). There is still some error due to the fact that the step-function underestimates the true amount that jobs in $\mathrm{OPT} \cap T_M$ cover on each time point. In the case of regular UFP-Cover, one can compensate this error by greedily selecting jobs that were not yet selected.



(a) Picking the blue part of the top job rules out Solution 1, while not picking it rules out Solution 2. Note that both solutions are incomparable since on t_1 and t_2 they cover different amounts of demand.



(b) Assume that the optimal solution selects all blue job parts (crossing t_M). Then there are still an exponential number of options for which jobs we should also select the green parts. Thus, we cannot take this decision immediately.

Figure 2 Locally Pareto-optimal choices.

In contrast to regular UFP-Cover, this approach fails for our generalization. We can think of each job as a collection of *parts* $[r_j, d_j^1), [d_j^1, d_j^2), ..., [d_j^{K-1}, d_j^K)$. The step function can only be guessed for the part that actually covers t_M. Yet, if we select that part, we need to pick all preceding parts of that job as well. This influences our options on the left side of t_M. On the other hand, if we do not pick the part that covers t_M of a certain job, succeeding parts of that job cannot be picked. This influences our options on the right side of t_M (see Figure 2a).

To address these issues we guess more fine-grained underestimating profiles. We group the jobs further such that for each job in a group the same part crosses t_M. Assume that for the jobs in the considered group their respective i-th part, $[d_j^{i-1}, d_j^i)$, covers t_M. We guess a *right underestimating profile* that estimates the total size covered to the right of t_M by these i-th parts that are selected by OPT. This profile partitions the jobs into subgroups according to the "step" of the profile in which the i-th part ends (see the green curve in Figure 1). For each of these constantly many groups we create a *subprofile* which underestimates the additional demand covered by the $(i+1)$-th parts of those jobs in OPT, i.e., by the intervals $[d_j^i, d_j^{i+1})$. We continue recursively and create underestimating subprofiles for all parts of the jobs, which gives a tree structure. We refer to this construction as *tree profiling*.

The tree profiling yields constantly many subgroups of jobs. For each of them we guess the number of jobs that the optimal solution selects (recall that the jobs in the same group have essentially the same size and cost). Then we recurse *only on the left* subproblem in which we want to cover the demand of the interval $[0, t_M)$ subject to the new constraint that from each subgroup we select the previously guessed number of jobs. Once we have a solution to this left subproblem, ideally we would like to decide how many parts of the jobs crossing t_M we select, i.e., the parts laying in the interval $[t_M + 1, T)$. Unfortunately, there can still be very many Pareto-optimal choices for this (see Figure 2b for an example). This can even happen when taking into account the information from the tree profiling. Instead, at this point we select for each job only the part that crosses t_M and we decide later about the additional parts we want to select. We recurse on the interval $[t_M + 1, T)$ and the remaining problem is to cover the demand of the interval $[t_M + 1, T)$ while we can select additional parts from the jobs that we selected already. In each subproblem we recurse on the respective middle time point, which yields a recursion depth of $O(\log n)$ and thus quasi-polynomial running time overall.

UFP-Cover for agreeable deadlines. Our second result is a pseudo-polynomial time $(1+\epsilon)$-approximation for UFP-Cover with agreeable deadlines. We first present an exact pseudo-polynomial time dynamic program (DP) for the case that the interval of each job is of the form $[0, d_j)$ or $[r_j, T)$, i.e., $r_j = 0$ or $d_j = T$. Then, we generalize this to the case where there are $1/\epsilon$ intervals $[T_0, T_1)$, $[T_1, T_2)$, ..., $[T_{1/\epsilon-1}, T_{1/\epsilon})$ and for each job j we have that $[r_j, d_j) \cap [T_\ell, T_{\ell+1})$ equals either $[r_j, T_{\ell+1})$ or $[T_\ell, d_j)$. Using the fact that the job deadlines are agreeable we can show that the time axis can be partitioned into a possibly superconstant number of intervals $[T_\ell, T_{\ell+1})$ with this property. By losing only a factor $1+\epsilon$ in the objective, we can split those into groups of at most $1/\epsilon$ consecutive intervals, each of which then yields an independent subinstance of our problem on which we apply our DP. The backbone of the latter is that the agreeable-deadlines property yields an ordering to process the jobs such that we need to remember only little information about the previously chosen jobs. We believe that this ordering and the resulting DP technique might be useful for other problems on agreeable intervals as well. Note that the opposite case where the job intervals form a laminar family has a simple exact DP. Thus, we can now handle the two "extreme" cases of the problem.

PTAS under resource augmentation. For UFP-Cover we present a pseudopolynomial time PTAS for the setting where we can enlarge each job interval $[r_j, d_j)$ by a factor $1+\mu$ for some $\mu > 0$, i.e., replace it by the interval $[r_j - \frac{\mu}{2}(d_j - r_j), d_j + \frac{\mu}{2}(d_j - r_j))$, while the compared optimal solution does not have this privilege. We use this resource augmentation to discretize the begin and end points of the intervals of the jobs. As in a similar result for UFP-packing [3], we group the jobs by the lengths of their intervals. In UFP-packing, the grouping can be done such that two jobs in different groups have intervals whose lengths differ by a large factor. Then each group can be handled almost independently. In our case we cannot establish such a property, since it requires the removal of some jobs from the input, which in turn may make our instance of UFP-Cover infeasible. Instead, our DP needs to transfer a lot of information between groups. The key for our approach is to prove that for each group it is sufficient to remember information from *one* previous group.

1.3 Other related work

The General Scheduling Problem can model a vast class of well-studied objective functions. The known $O(\log \log P)$-approximation for it [5], is even the best known result for several important special cases. For example, for the weighted flow time objective there were previously algorithms known with approximation ratios of $O(\log^2 P)$, $O(\log W)$ and $O(\log nP)$ [4, 10], where P and W denote the ranges of the job processing times and weights, respectively. Also, there is a QPTAS with a running time of $n^{O_\epsilon(\log P \log W)}$ [9].

For GSP with identical release dates the first constant factor approximation is due to Bansal and Pruhs [5] and yields an approximation ratio of 16. This was later improved to $4+\epsilon$ [11] by adapting ideas from the 4-approximation algorithm for UFP-Cover [6, 8]. For UFP-Cover this is the best known polynomial time result, while for quasi-polynomially bounded input numbers the problem even admits a QPTAS, implying a quasi-polynomial time $(e+\epsilon)$-approximation for GSP with identical release dates [12]. The used techniques are based on a QPTAS for the packing version of UFP [3]. For the latter algorithm, one can even remove the assumption that the input data is quasi-polynomially bounded [7]. The best known polynomial time results for UFP-packing are a $(2+\epsilon)$-approximation [1] and PTASs for some special cases [7].

2 QPTAS for GSP with identical release dates

We present our QPTAS for the General Scheduling Problem with identical release dates. Throughout this section we assume that all input numbers are quasi-polynomially bounded integers, and that we are given an $\epsilon > 0$ such that $1/\epsilon$ is an integer. We assume as well that we are given the number $f_{\max} = \max\{f_j(t) : f_j(t) \neq \infty, t \leq T\}$ as part of the input. First, we simplify the input such that the job cost functions attain only values that are powers of $1+\epsilon$ or ∞.

▶ **Lemma 1.** *By losing a factor $1+\epsilon$ in the objective, we can assume for each job j and each t that $f_j(t) \in \{(1+\epsilon)^k | k \in \mathbb{N}_0\} \cup \{0, \infty\}$ and that f_j is a non-decreasing step function with $O_\epsilon(poly(\log n))$ steps. We can further assume that each f_j is given explicitly, even if in the input it was given via an oracle.*

As in [5] we interpret GSP as a covering problem. Given a demand D_t for each interval $[t, t+1)$ and a set of jobs J. Each job $j \in J$ is characterized by a size p_j, a set of parts with corresponding intervals $I_j^1 = [t_j^{(0)}, t_j^{(1)}), I_j^2 = [t_j^{(1)}, t_j^{(2)})$, ... for $t_j^{(0)} \leq t_j^{(1)} \leq t_j^{(2)} \leq \dots$ and cost values $0 \leq c_j^1 < c_j^2 < \dots$. The goal is to select for each job j a prefix of its parts, i.e., a value $\sigma(j) \in \mathbb{N}_0$ such that all parts $k \leq \sigma(j)$ are selected. The cost for j is then $c_j^{\sigma(j)}$. Possibly $\sigma(j) = 0$ and then no part is selected, and thus we define $c_j^0 := 0$ for each job j. For a solution σ we say that a job j is *active* at time t if $t \in \cup_{i=1}^{\sigma(j)} I_j^i$. We require that for each t the total size of the jobs active at t is at least D_t, i.e., $\sum_{j : t \in \cup_{i=1}^{\sigma(j)} I_j^i} p_j \geq D_t$. The objective is to minimize the total cost $\sum_{j \in J} c_j^{\sigma(j)}$. We call this problem the *generalized UFP-Cover problem* (regular UFP-Cover is the special case where each job has only one part).

Using a similar argumentation as in [5] we can prove the following lemma.

▶ **Lemma 2.** *For any instance of GSP with identical release dates in which each cost function attains only polynomially many different values, we can construct in polynomial time an instance of generalized UFP-Cover such that approximations are preserved, i.e., for any $\alpha \geq 1$ an α-approximate solution for the generalized UFP-Cover instance can be transformed in polynomial time to an α-approximate solution for the GSP instance.*

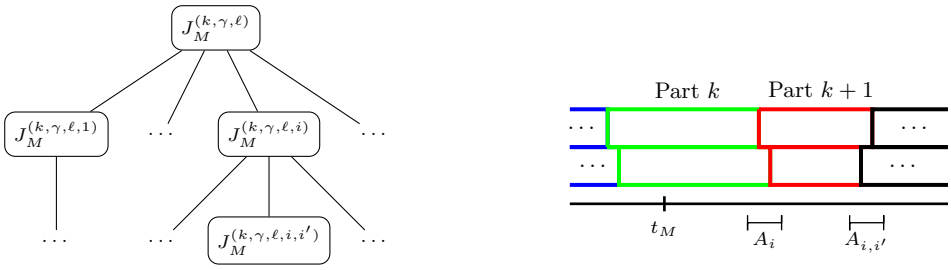
We apply Lemma 2 to reduce our given GSP instance to an instance of generalized UFP-Cover. Next, we ensure that each job has only $1/\epsilon^2$ parts.

▶ **Lemma 3.** *By losing a factor $1+\epsilon$ in the objective, we can assume that each job has at most $K := 1/\epsilon^2$ many parts, each value c_j^k is a power of $1+\epsilon$, and that $c_j^{k+1} = (1+\epsilon)c_j^k$ for each k.*

Assume w.l.o.g. that there is a value $T \leq \text{poly}(n)$ such that $I_j^k \subseteq [0, T)$ for each job j and each part k. Our algorithm is recursive. Let $t_M = \lceil \frac{T}{2} \rceil$ be the middle point of the interval $[0, T)$. The overall idea is to take a decision about the parts of jobs j that cover $[t_M, t_M + 1)$, i.e., such that $t_M \in I_j^k$ for some k, and then recursively decide on all job parts k' with $I_{j'}^{k'} \subseteq [0, t_M)$ (left subproblem) and k'' with $I_{j''}^{k''} \subseteq [t_M + 1, T)$ (right subproblem).

2.1 Tree profiling and grouping of jobs

Let $J_M \subseteq J$ denote the set of jobs j having a part k with $t_M \in I_j^k$. We partition J_M into a poly-logarithmic number of subsets according to their respective size, by the index of the



(a) Example of a tree $G^{(k,\gamma,\ell)}$. **(b)** Two jobs from group $J_M^{k,\gamma,\ell,i,i'}$.

Figure 3 Recursive partitioning of the jobs.

part that covers t_M, and by the cost of the latter. Formally, for numbers k, γ and λ we define sets

$$J_M^{(k,\gamma,\lambda)} := \{j \in J_M : t_M \in I_j^k,\ c_j^k = (1+\epsilon)^\gamma,\ \text{and } (1+\epsilon)^\lambda \le p_j < (1+\epsilon)^{\lambda+1}\}.$$

Consider one such group $J_M^{(k,\gamma,\lambda)}$. We want to partition it further. Let $\delta = \delta(\epsilon)$ be a small enough constant. First, we want to partition it into $O(1/\delta)$ subgroups $J_M^{(k,\gamma,\lambda,i)}$ such that: (i) OPT selects essentially the same number of jobs from each of these subgroups, and (ii) the k-th part of each job in the subgroup has a "similar" endpoint. Formally, we ensure the latter by partitioning the interval $[t_M, T)$ into subintervals $A_1, A_2, ...$ such that for each job j of a subgroup $J_M^{(k,\gamma,\lambda,i)}$ the k-th part ends in A_i (see Figure 3). Let $\bar{J}_M^{(k,\gamma,\lambda)}$ be the set of jobs $j \in J_M^{(k,\gamma,\lambda)} \cap OPT$ that OPT extends up to part k or further, i.e., for which OPT selects parts $I_j^1, ..., I_j^k$ and possibly more. To define our partition, we see that the respective k-th parts of the jobs in $\bar{J}_M^{(k,\gamma,\lambda)}$ cover some demand at each time point t, given by the function $\bar{f}_k(t) := \sum_{j \in \bar{J}_M^{(k,\gamma,\lambda)} : t \in I_j^k} p_j$. Observe that $\bar{f}_k$ is non-decreasing on $[0, t_M)$ and non-increasing on $[t_M, T)$. Ideally, we would like to guess $\bar{f}_k$ so that we have some idea about how much the k-th parts of the jobs in $J_M^{(k,\gamma,\lambda)}$ need to cover. Unfortunately, there are too many options on how $\bar{f}_k$ might look. Therefore, we guess $\left|\bar{J}_M^{(k,\gamma,\lambda)}\right|$ and a simpler underestimating function $\tilde{f}_k$ that approximates $\bar{f}_k$ sufficiently well, as given by the following lemma (see Figure 1). For our later purposes we need this function only on the interval $[t_M, T)$.

▶ **Lemma 4.** *There exists a function $\tilde{f}_k : [t_M, T) \to \{0, 1, ..., \sum_{j \in J} p_j\}$ such that $\tilde{f}_k$ is a step-function with at most $O(1/\delta)$ many steps, $\tilde{f}_k(t) \le \bar{f}_k(t) \le \tilde{f}_k(t) + \delta \cdot \left|\bar{J}_M^{(k,\gamma,\lambda)}\right| \cdot (1+\epsilon)^{\lambda+1}$, and $\tilde{f}$ is non-increasing.*

We use the function $\tilde{f}_k$ to split the set $J_M^{(k,\gamma,\lambda)}$ into subgroups, according to where the part k of each job $j \in J_M^{(k,\gamma,\lambda)}$ ends. Let $A_1, A_2, ...$ denote a partition of $[t_M, T)$ into $O(1/\delta)$ subintervals such that on each subinterval A_i the function $\tilde{f}_k$ is constant. For each such interval A_i we define $J_M^{(k,\gamma,\lambda,i)} \subseteq J_M^{(k,\gamma,\lambda)}$ to be the jobs $j \in J_M^{(k,\gamma,\lambda)}$ such that $(t_j^{(k)} - 1) \in A_i$ (recall that $I_j^k = [t_j^{(k-1)}, t_j^{(k)})$).

Subprofiles. It is convenient to think of a tree where $J_M^{(k,\gamma,\lambda)}$ forms the root node and the sets $J_M^{(k,\gamma,\lambda,i)}$ form the children of $J_M^{(k,\gamma,\lambda)}$. We take each such group $J_M^{(k,\gamma,\lambda,i)}$ and partition it further into $O(1/\delta)$ smaller subgroups $J_M^{(k,\gamma,\lambda,i,1)}, J_M^{(k,\gamma,\lambda,i,2)}, ...$ In the tree view, we can think of appending those as children to the node for the group $J_M^{(k,\gamma,\lambda,i)}$, see Figure 3. For

the subgroups, as before our goal is that OPT selects essentially the same number of jobs from each subgroup $J_M^{(k,\gamma,\lambda,i,i')}$ and that for each such subgroup the $(k+1)$-th part has a "similar" end point.

Recall that when we partitioned $J_M^{(k,\gamma,\lambda)}$ we estimated what the k-th part of the jobs in $J_M^{(k,\gamma,\lambda)} \cap OPT$ cover (via the function $\tilde{f}_k$) and obtained a grouping according to the steps of $\tilde{f}_k$. For the finer partitioning of $J_M^{(k,\gamma,\lambda,i)}$ we consider the jobs in $J_M^{(k,\gamma,\lambda,i)}$ for which OPT selects also the $(k+1)$-th part (and thus also the k-th part). Denote that set as $\bar{J}_M^{(k,\gamma,\lambda,i)}$. We define the function $\bar{f}_{k,i}(t)$ that models how much the k-th and the $(k+1)$-th parts of the jobs in $\bar{J}_M^{(k,\gamma,\lambda,i)} \cap OPT$ cover. Formally, $\bar{f}_{k,i}(t) := \sum_{j \in \bar{J}_M^{(k,\gamma,\lambda,i)} : t \in I_j^k \cup I_j^{k+1}} p_j$. We guess an underestimating function $\tilde{f}_{k,i}$ with the same properties as the function $\tilde{f}_k$ as given in Lemma 4, i.e., $\tilde{f}_{k,i}$ has $O(1/\delta)$ many steps, $\tilde{f}_{k,i}(t) \leq \bar{f}_{k,i}(t) \leq \tilde{f}_{k,i}(t) + O(\delta) \cdot |\bar{J}_M^{(k,\gamma,\lambda,i)}| \cdot (1+\epsilon)^{\lambda+1}$, and $\tilde{f}_{k,i}$ is non-increasing. Like before, the steps of $\tilde{f}_{k,i}$ yield a partition of $[t_M, T)$ into $O(1/\delta)$ many subintervals $A_{i,1}, A_{i,2}, \ldots$ such that $\tilde{f}_{k,i}$ is constant in each of them. Each such subinterval $A_{i,i'}$ yields a subgroup $J_M^{(k,\gamma,\lambda,i,i')}$ that contains all jobs $j \in J_M^{(k,\gamma,\lambda,i)}$ whose $(k+1)$-th part ends in $A_{i,i'}$, i.e., $(t_j^{(k+1)} - 1) \in A_{i,i'}$.

We continue recursively for K levels, expanding the tree accordingly. Analogous to before, we obtain for each node v in level k' of the tree (that is not a leaf), a subprofile function $\bar{f}_v$ and an approximate version $\tilde{f}_v$ such that $\tilde{f}_v(t) \leq \bar{f}_v(t) \leq \tilde{f}_v(t) + O(\delta) \cdot |\bar{J}_M^{(k,\gamma,\lambda,v)}| \cdot (1+\epsilon)^{\lambda+1}$, where $\bar{J}_M^{(k,\gamma,\lambda,v)}$ is the set of jobs in J_v that the optimum extends up to its $(k+k')$-th part. The leafs of the tree yield a partition of the job set, and the total number of nodes is $(1/\delta)^K$.

We can guess the whole partition in time $n^{(1/\delta)^{O(K)}}$ which will eventually be bounded by $n^{O_\epsilon(1)}$ (note that there are only $T \leq \text{poly}(n)$ options for each endpoint of an interval A_i or $A_{i,i'}$, etc.). In the same running time, we can guess for each arising group and subgroup the total number of jobs that OPT selects from these groups. More precisely, let $G^{(k,\gamma,\lambda)}$ be the resulting tree and for each node v denote by J_v the corresponding job group. For a node v on level k' we guess the value $N(v)$, the number of jobs in J_v that the optimum extends at least up to their respective $(k+k')$-th part.

We now bound the total demand deficit made by the underestimating functions. Let $f(t) = \sum_{j \in J_M^{(k,\gamma,\lambda)} : j \text{ active at } t \text{ in OPT}} p_j$ be the total size of jobs in $J_M^{(k,\gamma,\lambda)}$ that cover t in the optimal solution. We say that a solution is *concordant* with the tree $G^{(k,\gamma,\lambda)}$ and numbers $N(v)$ if, for each node v of each level k', the solution selects the $(k+k')$-th part of at least $(1+\epsilon)N(v)$ jobs in J_v, or of all jobs in J_v in case that $|J_v| < (1+\epsilon)N(v)$. As the next lemma shows, any tree concordant solution covers the demand at any point t almost as good as the optimal solution. The gap is bounded by $K \cdot \delta \cdot |\bar{J}_M^{(k,\gamma,\lambda)}| \cdot (1+\epsilon)^{\lambda+1}$ which is an upper bound on the sum of the deficits of all subprofiles relevant for a time point t. Here $\bar{J}_M^{(k,\gamma,\lambda)}$ is the subset of jobs in $J_M^{(k,\gamma,\lambda)}$ that OPT extends at least to the k-th part.

▶ **Lemma 5.** *Consider any solution concordant with tree $G^{(k,\gamma,\lambda)}$. The demand covered by such a solution at any time $t \in [t_M, T)$ is at least $f(t) - K \cdot \delta \cdot |\bar{J}_M^{(k,\gamma,\lambda)}| \cdot (1+\epsilon)^{\lambda+1}$.*

2.2 Fixing the demand deficit

We would like to recurse on the left and on the right subproblem, i.e., on $[0, t_M)$ and $[t_M + 1, T)$. We have guessed the correct number of jobs in each group but we have not decided yet which exact jobs from each group we want to select.

We deal with these issues as follows. Let us fix a tree $G^{(k,\gamma,\lambda)}$. We first consider any solution ALG that is concordant with the tree. By Lemma 5, this solution already covers almost all necessary demand, having a deficit of at most $\delta \cdot |\bar{J}_M^{(k,\gamma,\lambda)}| \cdot (1+\epsilon)^{\lambda+1}$ for every

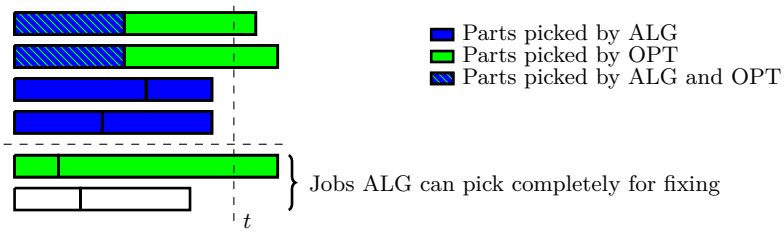


Figure 4 In contrast to the regular UFP-Cover Problem where selecting new jos is always sufficient, here this is not the case: even selecting all the new (bottom) jobs does not suffice to cover t! Instead, extending previously selected jobs is necessary.

time point in $[t_M, T]$. Even if we can fix this demand by adding more jobs (and we will, essentially, do so), picking an arbitrary concordant solution at this point will create issues for the left subproblem: nothing guarantees that the chosen solution we pick allows to cover the remaining demand within $[0, t_M)$ at a reasonable cost. To avoid this problem, we call the left subproblem recursively, giving the trees $G^{(k,\gamma,\lambda)}$ and numbers $N(v)$ for each node as input. We will require this problem to give us a solution ALG that is concordant with the tree for each group $J_M^{(k,\gamma,\lambda)}$ and that satisfies all demand at $[0, t_M)$. The exact way of solving this left subproblem is given in the next subsection. We call a solution constructed this way a *left-feasible* solution.

Consider now a left-feasible solution ALG and fix a tree $G^{(k,\gamma,\lambda)}$. The idea is to fix the deficit in $[t_M, T)$ by adding jobs picked greedily. As a first approach we could consider the following method: within all jobs in $J_M^{(k,\gamma,\lambda)}$ not active at t_M, pick the $\delta|\bar{J}_M^{(k,\gamma,\lambda)}|(1+\epsilon)$ ones that extend furthest to the right when all of their K parts are chosen. We denote by $H^{(k,\gamma,\lambda)}$ the set of these jobs. For any timepoint t that is covered by all these jobs, we will cover the whole deficit. Also, we can show that total incurred cost is at most an ϵ-fraction of the cost of $\mathrm{OPT} \cap \bar{J}_M^{(k,\gamma,\lambda)}$. One might be tempted to conclude that we are done: since we picked the jobs greedily, a time point t that is not covered by all jobs in $H^{(k,\gamma,\lambda)}$ cannot be covered by any other job that we did not make active at t_M. This is indeed enough to argue in the regular UFP-Cover problem [12]. However, the argument fails in our setting as we might still be able to further extend some jobs that our solution picks to cover t_M but not are not extended to cover t; see Figure 4.

To solve this issue we truncate ALG by removing for each group $J_M^{(k,\gamma,\lambda)}$ and each job $j \in J_M^{(k,\gamma,\lambda)}$ all parts that do not cover any point $t \in [0, t_M + 1)$. Let ALG_M be the truncated solution. We show that ALG_M plus all parts of all jobs in $H^{(k,\gamma,\lambda)}$ can be extended (by adding more parts, not necessarily like ALG) to a solution that covers all required demand and costs at most a $1+\epsilon$ factor more than OPT. The constructed solution covers all demand at times $[0, t_M + 1)$ and we will solve the remaining problem in the right subproblem.

To make this idea formal, denote by OPT_M the solution obtained by taking OPT and removing from it all parts I_j^k such that $I_j^k \subseteq [t_M + 1, T)$. For any left-feasible solution S we say that a solution S' is an *extension* of S if for each job j the solution S' extends j up to at least as many parts as S.

▶ **Lemma 6.** *Assume that $1/\delta = K \cdot \epsilon(1+\epsilon)^{O(K)}$. Suppose we are given the left-feasible truncated solution ALG_M. Then we can compute in polynomial time a set of jobs $H \subseteq J_M$ such that*

- *if we select all parts of each job in H this yields a total cost of at most $O(\epsilon) \cdot c(\mathrm{OPT}_M)$, and*
- *for the solution $ALG_M \cup H$ there is an extension ALG' such that $c(ALG') - c(ALG_M \cup H) \leq c(\mathrm{OPT}) - c(\mathrm{OPT}_M)$.*

Proof Sketch. Consider a set $J_M^{(k,\gamma,\lambda)}$. We consider all jobs of this set that are not covering t_M in ALG and sort them non-increasingly with respect to the right endpoint of I_j^K. Let $H^{(k,\gamma,\lambda)}$ be the set of the first $K \cdot \delta |\bar{J}_M^{(k,\gamma,\lambda)}|(1+\epsilon)$ such jobs and define $H = \cup_{k,\gamma,\lambda} H^{(k,\gamma,\lambda)}$. Notice that extending all parts of jobs in $H^{(k,\gamma,\lambda)}$ incurs a cost of at most $K \cdot \delta(1+\epsilon)^{K+1}(1+\epsilon)^{\gamma}|\bar{J}_M^{(k,\gamma,\lambda)}|$. By choosing the constants in the definition of δ appropriately we obtain that the cost is $O(\epsilon) \cdot c(\bar{J}_M^{(k,\gamma,\lambda)})$. Summing over all triplets k, γ, λ yields the desired bound on the total cost of H.

For a given set $H^{(k,\gamma,\lambda)}$, out of all right endpoints of jobs in the set, call τ_R the one most to the left. Inside the interval $[t_M, \tau_R)$ the jobs in $H^{(k,\gamma,\lambda)}$ cover at least $K(1+\epsilon)^{\lambda+1}\delta|\bar{J}_M^{(k,\gamma,\lambda)}|$, and thus they cover all deficit left by the solution ALG (or any other tree concordant solution). On the other hand, for any $t > \tau_R$ our greedy choice for $H^{(k,\gamma,\lambda)}$ guarantees that *all* jobs in $J_M^{(k,\gamma,\lambda)}$ that can be extended to cover t are taken at least up to their k-th part in $ALG_M \cup H$. This allows us to construct the claimed extension ALG' of $ALG_M \cup H$: we start with ALG and transform it step by step to make it resemble the optimal solution. Note that this is a purely existential result since we need to know the optimal solution for this procedure. ◀

2.3 Left subproblem

Suppose that via recursion we have computed a left-feasible solution ALG. Then, using Lemma 6 we compute the jobs H such that $c(H) \leq O(\epsilon) \cdot c(OPT_M)$ and such that the extension ALG' is guaranteed to exist. In order to compute (an approximation to) ALG' we recurse on the right subproblem, given by the interval $[t_M + 1, T)$.

For each $t \in [t_M + 1, T)$ we update the demand D_t to take into account that we already selected some job parts crossing t_M and the jobs in H. Formally, we define the new demands as $D'_t := D_t - \sum_{j:t\in\tilde{J}(t)} p_j$ where $\tilde{J}(t)$ denotes the set of jobs j such that $ALG_M \cup H$ contains a part of j that covers t. For each job j such that ALG_M selected the part I_j^k covering t_M, our subproblem only has the parts of j that lie completely within $[t_M + 1, T)$. We update their cost, taking into account that the left subproblem has already paid c_j^k for it, i.e., the cost value $\bar{c}_j^{\ell-k}$ for each new part $\ell - k$ is set to $\bar{c}_j^{\ell-k} = c_j^{\ell} - c_j^k$. This yields an instance of our problem on the interval $[t_M + 1, T)$ whose size is only half the size of the original interval $[0, T)$. Strictly speaking, the new costs might no longer be a power of $1 + \epsilon$. However, note that the adjustment of costs means that $\bar{c}_j^1 = c_j^{k+1} - c_j^k = \epsilon c_j^1$. Therefore, the costs of any two parts of a job still differ by at most a constant factor and the new cost values come from a set of size $O(\text{poly}(\log n))$ (which is important to bound the number of job groups $J_M^{(k,\gamma,\lambda)}$) . Moreover, this factor does not increase further in the recursion and hence we can recurse one the right subproblem with essentially the same routine as above.

It remains to describe how to recurse on the left subproblem for the interval $[0, t_M)$. Formally, this subproblem is defined as follows: we are given the interval $[0, t_M)$ together with the demand D'_t for each point $t \in [0, t_M)$ (the updated demand). Also, for each tree $G^{(k,\gamma,\lambda)}$ and each vertex v we are given a corresponding group of jobs J_v. Additionally we have to consider the set of all input jobs j such that no part of j crosses t_M - we refer to this set of jobs as J_L. Finally, for each group J_v we are given a value $N(v)$ that indicates that for at least $(1 + \epsilon)N(v)$ jobs in J_v we have to select the respective part that crosses t_M.

Our objective is to find a solution for jobs in $J_L \cup \bigcup_{k,\gamma,\lambda} J^{(k,\gamma,\lambda)}$ that covers all demand in $[0, t_M)$ and that is concordant for each tree. To have a cleaner subproblem, we observe that the leaves of $G^{(k,\gamma,\lambda)}$ imply a *partition* of the jobs of $J^{(k,\gamma,\lambda)}$ into subgroups. For each of them we guess how many jobs the optimal solution selects from the subset corresponding to that leaf. Then for each of them we require that the left subproblem selects either a factor

$1+\epsilon$ more jobs or all jobs from that subset. The resulting solution can easily be transformed into a concordant solution.

We consider the point $t'_M := \lceil \frac{t_M}{2} \rceil$. Like above, we partition the jobs into groups according to which part of them crosses t'_M. However, we do this separately for J_L and each subgroup of jobs crossing t_M. For each resulting separate group, we guess the profiles and recursive subprofiles as before. Once we have guessed this partition of the jobs together with the required number of jobs of each group, we recurse on the left-left problem, i.e., on the problem for the interval $[0, t'_M)$. When we obtained a solution for the left-left subproblem in the interval $[0, t'_M)$ we recurse further on the interval $[t'_M, t_M)$. For this left-right subproblem, we update the cost of the jobs whose respective parts crossing t'_M were selected by the left-left subproblem (like we did when we defined the right subproblem of the interval $[t_M + 1, T)$) and additionally impose the constraint that from each group J_v (as defined by the main subproblem for the interval $[0, T)$) for at least $(1+\epsilon)N(v)$ jobs we select the respective part crossing t_M.

Number of groups. We continue recursively in the same fashion. In the recursion, the number of job groups we pass to each subproblem increases since from the main subproblem for the interval $[0, T)$ we are given a partition into subgroups and whenever we recurse on a left subproblem these subgroups are partitioned further and also new subgroups are defined. However, we can show that in each step of the recursion the total number of arising subgroups is bounded by $(\frac{1}{\epsilon^2} \log n)^{O(K)}$.

▶ **Lemma 7.** *In the input of each subproblem arising in the recursion, the jobs are partitioned into at most* $(\frac{1}{\epsilon^2} \log n)^{O(K)}$ *different groups.*

Whenever we are given a subproblem on some interval I' then we guess subgroups and certain values with a quasi-polynomial number of options in total and we recurse on two subproblems, given by subintervals of I' whose size is half the size of I'. Thus, the recursion tree has a depth of $O(\log T) = O(\log n)$ and each internal node of the tree has at most quasi-polynomially many children. Hence, our algorithm has quasi-polynomial running time overall.

▶ **Theorem 8.** *There are quasi-polynomial time* $(1+\epsilon)$*-approximation algorithms for the general scheduling problem and for the generalized UFP-Cover problem, assuming that all input values are quasi-polynomially bounded integers.*

3 Agreeable Instances

In this section we present our pseudopolynomial-time $(1+\epsilon)$-approximation algorithm for the UFP-Cover problem on agreeable instances. We first show how to partition a given instance into smaller subinstances (Section 3.1). Then we then present our algorithm for a special type of subinstances (Section 3.2).

For simplicity of presentation, we will assume throughout this section w.l.o.g. that each integer timepoint t is associated with a demand D_t and that we only need to cover the demands at such timepoints. Furthermore, we assume w.l.o.g. that the the intervals defined by the release-time and deadline of each job are closed, i.e., have the form $[r_j, d_j]$. We also assume that all elements of the set $U := \cup_j \{r_j, d_j\}$ are disjoint. To simplify the presentation, we further assume w.l.o.g. that all elements of U are even integers.

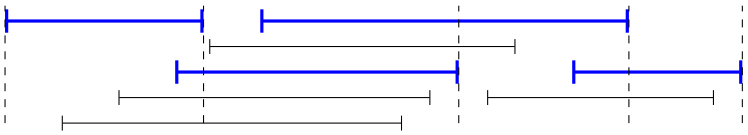


Figure 5 The thick blue jobs are the pivotal jobs, and the dashed vertical lines define the intervals. It is helpful to think of r_j for the first pivotal job j as the start point of the first interval.

3.1 Preprocessing & Preliminary Observations

We partition the time-horizon into intervals. We may assume that there is no timepoint throughout $[0, T]$ that is not covered by any job, since then we could easily seperate the instance at this timepoint into two independent subinstances. For our partitioning we inductively introduce a set of *pivotal jobs* P.

▶ **Definition 9.** The first pivotal job is the job with earliest start time r_j. We define the other pivotal jobs by induction. Assume that we have defined the first k pivotal jobs $j_1, ..., j_k$. Then the $(k+1)$-th pivotal job is the job with latest start time among all jobs j with $r_j \leq d_{j_k}$.

We use the end points of the pivotal jobs in order to partition the time horizon into intervals $\mathcal{I}$. More formally, we partition the time horizon into intervals at timepoints $X := \{d_j : j \in P\} \cup \{0\}$. Let $T_0, T_1, \cdots$ be the timepoints in X in increasing order. Then each interval in $\mathcal{I}$ is of the form $[T_k, T_{k+1}]$ for some $k \in \mathbb{N}$. See Figure 5 for an example.

▶ **Lemma 10.** *The $[r_j, d_j]$-interval of any non-pivotal job intersects at most one timepoint of X. The $[r_j, d_j]$-interval of any pivotal job intersects at most two timepoints of X.*

Next, we cut the instance into subinstances so that each subinstance contains at most q many intervals (in our final algorithm we will set $q = O(1/\epsilon)$). We do this in a randomized fashion but the procedure can be easily derandomized, similar to, e.g., [2]. Let x be a random variable that takes its value uniformly at random among the integers $\{0, 1, 2, \ldots, q-1\}$. We "cut" the instance into subinstances at timepoints $W := \{T_x, T_{x+q}, T_{x+2q}, \ldots\}$. Let each subinstance $I_i := [T_{x+iq}, T_{x+(i+1)q}]$ contain all jobs j whose interval $[r_j, d_j]$ intersects I_i. Jobs j whose intervals $[r_j, d_j]$ span two consecutive subinstances I_i and I_{i+1} are split into two jobs: a job j' with $r_{j'} := r_j, d_{j'} := T_{x+iq}, p_{j'} := p_j, c_{j'} := c_j$, and a job j'' with $r_{j''} := T_{x+iq}, d_{j''} := d_j, p_{j''} := p_j, c_{j''} := c_j$.

Note that the choice of x can be derandomized by trying out all q possible choices for x and selecting the best one. For the obtained division into subinstances we prove the following lemma.

▶ **Lemma 11.** *An exact algorithm with running time $O(f(n))$ for a subinstance containing at most q consecutive intervals from $\mathcal{I}$ yields a $(1 + 2/q)$-approximation algorithm for the original instance, with a running time of $O(n \cdot f(n))$.*

We give a pseudopolynomial-time exact algorithm for the problem on instances with $q = O(1/\epsilon)$ many consecutive intervals. The algorithm is based on dynamic programming. Due to space constraints in the main body of the paper we only present a simpler version of our DP for the case $q = 1$ in Subsection 3.2.

3.2 Solving a subinstance with only one interval

Assume that we are given a subinstance consisting of only one interval I_i. We form a partition $J_L \dot{\cup} J_R$ for the jobs whose $[r_j, d_j]$-interval intersects this interval I_i. The set J_L is the set of

all jobs j such that $d_j \in I_i$, and J_R is the set of such jobs j such that $r_j \in I_i$. By Lemma 10, $J_L \dot\cup J_R$ comprises the whole set of jobs j such that $[r_j, d_j] \cap I \neq \emptyset$. We now define a set of relevant timepoints M for our interval I_i as $M := (U \cap I_i)$, where $U := \{r_j, d_j | j \in J\}$ is the set of all globally relevant timepoints.

Let us consider this set ordered from left to right, so that $M = \{t_1, t_2 \dots t_k\}$. We fill out the table of our dynamic program in a bottom-up fashion by considering these timepoints in reverse order, that is from right to left. Each cell of the dynamic programming table has the form $T[t_z, b, i_L, c_L, c_R]$. Intuitively, it describes the subproblem of covering the demand on the subinterval $[t_z, t_k]$ by a set of jobs $J'_L \subseteq J_L$ having their respective deadline in $[t_z, t_k + 1]$ with $p(J'_L) := \sum_{j \in J'_L} p_j = i_L$ and $c(J'_L) := \sum_{j \in J'_L} c_j = c_L$, and by a set of jobs $J'_R \subseteq J_R$ having their respective release dates in $[t_z, t_k + 1]$ with $c(J'_R) = c_R$. The demand at each point $t \in [t_z, t_k]$ is $D_t - b$, i.e., the reader may imagine that some other routine of the global algorithm selects jobs with a total size of b that cover each point in $[t_z, t_k]$.

Formally, this DP cell is filled out with a "yes" if and only if there exist two sets $J'_L \subseteq J_L$ and $J'_R \subseteq J_R$, such that:

(i) for each job $j \in J'_R$, there holds $r_j \geq t_z$, and for each job $j \in J'_L$ there holds $d_j \geq t_z$,
(ii) $p(J'_L) = \sum_{j \in J'_L} p_j = i_L$ and $c(J'_L) = \sum_{j \in J'_L} c_j = c_L$,
(iii) $c(J'_R) = c_R$, and
(iv) $\forall \ell : z \leq \ell \leq k, \sum_{j \in J'_L \cup J'_R : [r_j, d_j] \ni t_\ell} p_j \geq D_{t_\ell} - b$.

Filling out the table. We fill out the table starting with all entries for the rightmost timepoint t_k. First, we fill in $T[t_k, b, i_L, c_L, c_R]$ for all possible values of $0 \leq i_L \leq \sum_j p_j$, $0 \leq c_L, c_R \leq \sum_j c_j$, and $0 \leq b \leq \sum_{j \in J_R} p_j$. Note that for such a cell only the pivotal job j_p of the interval is relevant since no other job can have its release date or deadline at t_k. For filling in the entry it suffices to consider the two possibilities of selecting j_p and not selecting j_p.

Assume now that we have filled in all cells corresponding to timepoints from t_{z+1} to t_k and we want to fill in the entries for t_z. The timepoint t_z is the start or the end point of a job j that either belongs to J_L or to J_R. The entries for t_z in our dynamic programming table depend on the set to which j belongs to, and on whether j is added to the solution. Formally, if $j \in J_R$, then $T[t_z, b, i_L, c_L, c_R]$ = "yes" if and only if $T[t_{z+1}, b, i_L, c_L, c_R]$ = "yes" and $i_L + b \geq D_{t_z}$ or if $T[t_{z+1}, b + p_j, i_L, c_L, c_R - c_j]$ = "yes" and $i_L + b + p_j \geq D_{t_z}$. So either we do not add j to the solution, and then we need to cover the demand at t_z with the jobs already selected for t_{z+1}, or we add j to the solution, and then we can add its size to the respective b-entry at t_{z+1}. Symmetrically, if $j \in J_L$, then $T[t_z, b, i_L, c_L, c_R]$ = "yes" if and only if we have that $T[t_{z+1}, b, i_L, c_L, c_R]$ = "yes" and $i_L + b \geq D_{t_z}$ or if $T[t_{z+1}, b, i_L - p_j, c_L - c_j, c_R]$ = "yes" and $i_L + b \geq D_{t_z}$.

By keeping track of the respective sets J_L and J_R in each cell we are able to reconstruct our solution starting from the cell of the form $T[t_1, 0, i_L, c_L, c_R]$ that minimizes $c_L + c_R$ among all such cells with a "yes"-entry. Our dynamic program requires pseudopolynomial running time, because the considered possible values for c_L, c_R, i_L and b are pseudopolynomial in the input size. It returns an exact solution to the given problem. We are able to generalize these ideas to subinstances with $O(1/\epsilon)$ many intervals, and thus prove the following theorem.

▶ **Theorem 12.** *There is a pseudopolynomial-time $(1 + \epsilon)$-approximation algorithm for the UFP-Cover problem on agreeable instances.*

References

1. Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing 2+ϵ approximation for unsplittable flow on a path. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 26–41, 2014.
2. Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, January 1994. doi:10.1145/174644.174650.
3. Nikhil Bansal, Amit Chakrabarti, Amir Epstein, and Baruch Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC 2006)*, pages 721–729. ACM, 2006. doi:10.1145/1132516.1132617.
4. Nikhil Bansal and Kedar Dhamdhere. Minimizing weighted flow time. *ACM Transactions on Algorithms*, 3(4):Article 39, 2007. doi:10.1145/1290672.1290676.
5. Nikhil Bansal and Kirk Pruhs. The geometry of scheduling. *SIAM Journal on Computing*, 43(5):1684–1698, 2014. doi:10.1137/130911317.
6. Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001. doi:10.1145/502102.502107.
7. Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 47–58. SIAM, 2015. doi:10.1137/1.9781611973730.5.
8. Venkatesan T. Chakaravarthy, Amit Kumar, Sambuddha Roy, and Yogish Sabharwal. Resource allocation for covering time varying demands. In *Algorithms–ESA 2011*, pages 543–554. Springer, 2011. doi:10.1007/978-3-642-23719-5_46.
9. Chandra Chekuri and Sanjeev Khanna. Approximation schemes for preemptive weighted flow time. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 297–305. ACM, 2002. doi:10.1145/509907.509954.
10. Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 84–93, 2001. doi:10.1145/380752.380778.
11. Maurice Cheung, Julián Mestre, David B. Shmoys, and José Verschae. A primal-dual approximation algorithm for min-sum single-machine scheduling problems. *SIAM Journal on Discrete Mathematics. To appear*, 2017.
12. Wiebke Höhn, Julián Mestre, and Andreas Wiese. How unsplittable-flow-covering helps scheduling with job-dependent cost functions. In *International Colloquium on Automata, Languages, and Programming*, pages 625–636. Springer, 2014. doi:10.1007/978-3-662-43948-7_52.
13. Eugene L. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977. doi:10.1016/S0167-5060(08)70742-8.

Improved Algorithms for MST and Metric-TSP Interdiction*†

André Linhares[1] and Chaitanya Swamy[2]

1 **Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON, Canada**
alinhare@uwaterloo.ca
2 **Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON, Canada**
cswamy@uwaterloo.ca

Abstract

We consider the *MST-interdiction* problem: given a multigraph $G = (V, E)$, edge weights $\{w_e \geq 0\}_{e \in E}$, interdiction costs $\{c_e \geq 0\}_{e \in E}$, and an interdiction budget $B \geq 0$, the goal is to remove a set $R \subseteq E$ of edges of total interdiction cost at most B so as to maximize the w-weight of an MST of $G - R := (V, E \setminus R)$.

Our main result is a 4-approximation algorithm for this problem. This improves upon the previous-best 14-approximation [31]. Notably, our analysis is also significantly simpler and cleaner than the one in [31]. Whereas [31] uses a greedy algorithm with an involved analysis to extract a good interdiction set from an over-budget set, we utilize a generalization of knapsack called the *tree knapsack problem* that nicely captures the key combinatorial aspects of this "extraction problem." We prove a simple, yet strong, LP-relative approximation bound for tree knapsack, which leads to our improved guarantees for MST interdiction. Our algorithm and analysis are nearly tight, as we show that one cannot achieve an approximation ratio better than 3 relative to the upper bound used in our analysis (and the one in [31]).

Our guarantee for MST-interdiction yields an 8-approximation for *metric-TSP interdiction* (improving over the 28-approximation in [31]). We also show that *maximum-spanning-tree interdiction* is at least as hard to approximate as the minimization version of densest-k-subgraph.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.1.6 Optimization, G.2 Discrete Mathematics

Keywords and phrases Approximation algorithms, interdiction problems, LP-rounding algorithms, iterative rounding, tree-knapsack problem, supermodular functions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.32

1 Introduction

Interdiction problems are a broad class of optimization problems with a wide range of applications. They model the problem faced by an attacker, who given an underlying, say, minimization, problem, aims to destroy or *interdict* the elements involved in the optimization problem (e.g., nodes or edges in a network-optimization problem) without exceeding a given interdiction budget, so as to maximize the optimal value of the residual optimization problem (where one cannot use the interdicted elements). A classical example is the *minimum-spanning-tree (MST) interdiction* problem [22, 9, 31], which is the focus of this work: we are

* A full version of the paper is available at https://arxiv.org/abs/1706.00034.
† This work was supported in part by NSERC grant 327620-09 and an NSERC Discovery Accelerator Supplement Award.

given a multigraph $G = (V, E)$, edge weights $\{w_e \geq 0\}_{e \in E}$, interdiction costs $\{c_e \geq 0\}_{e \in E}$, and an interdiction budget $B \geq 0$; the goal is to interdict (i.e., remove) a set $R \subseteq E$ of edges of total interdiction cost at most B so as to maximize the w-weight of an MST of the multigraph $G - R := (V, E \setminus R)$. Note that G may have parallel edges, which can be useful in modeling partial-interdiction effects, wherein interdicting an edge causes an increase in its weight that depends on the interdiction cost incurred for the edge.

At a high level, interdiction problems can be seen as investigating the sensitivity of an underlying optimization problem with respect to the removal of a limited set of underlying elements. This type of sensitivity analysis may be utilized to identify vulnerable spots (e.g., regions in a network) either: (a) for possible reinforcement, or, (b) if the optimization problem models an undesirable process (e.g., the spread of infection, or nuclear-arms smuggling), for disruption, so as to maximally impair the underlying process. A variety of applications of interdiction problems ensue from these two perspectives, including infrastructure protection [5, 27], hospital-infection control [1], prevention of nuclear-arms smuggling [24], and military planning [11] (see also the references in [31]). Consequently, interdiction problems have been extensively studied, especially in the Operations Research literature; besides MST-interdiction, some well-studied interdiction problems include network-flow interdiction [25, 28, 2, 30, 13, 3], shortest s-t path interdiction [10, 14, 17, 20], and maximum-matching interdiction [29, 6]. All these problems, as well as MST-interdiction, are *NP*-hard.

Our results. Our main result is a 4-approximation algorithm for MST interdiction (Theorem 7), i.e., we compute in polytime a solution of value at least (optimum)/4. This is a substantial improvement over the previous-best approximation ratio of 14 obtained by Zenklusen [31].

Notably, and perhaps more importantly, our algorithm is simple, and its analysis is significantly simpler and cleaner than the one in [31]. The key ingredient (see also "Our techniques") of both our algorithm and the one in [31] is a procedure for extracting a good interdiction set from one that exceeds the interdiction budget. Whereas [31] uses a greedy algorithm with a rather involved analysis to achieve this, our simple and more-effective procedure is based on two chief insights. First, we discern that the key combinatorial aspects of this "extraction problem" can be captured quite nicely via a clean generalization of the knapsack problem called the *tree knapsack problem* [15] (Section 3). In particular, we argue that approximation guarantees for tree knapsack *relative to the natural LP for this problem* translate directly to guarantees for MST interdiction. Second, complementing the above insight, we show that the tree knapsack problem admits a simple *iterative-rounding* based algorithm that achieves a strong LP-relative guarantee (Theorem 4, Corollary 6). Our improved guarantee for MST interdiction then readily follows by combining these two ideas.

We also show a lower bound of 3 (Theorem 16) on the approximation ratio achievable relative to the upper bound used in our analysis (and the analysis in [31]), thereby showing that our algorithm and analysis are nearly tight.

Our MST-interdiction result also yields an improved guarantee for the *metric-TSP interdiction* problem (Section 5): given *metric* edge weights $\{w_e\}$, we now seek an interdiction set R with $\sum_{e \in R} c_e \leq B$ so as to maximize the minimum w-weight of a closed walk in $G - R$ that visits all nodes at least once. Since an α-approximation for MST interdiction yields a 2α-approximation for metric-TSP interdiction [31], we obtain an approximation factor of 8 for metric-TSP interdiction, which improves upon the previous-best factor of 28 [31].

In Section 6, we consider the *maximum-spanning-tree interdiction* problem, where the goal is to *minimize the maximum w-weight of a spanning tree of $G - R$*. We show that this problem is at least as hard to approximate as the minimization version of the *densest-k-subgraph* problem (MinDkS). MinDkS does not admit any constant-factor approximation under certain less-standard complexity assumptions [26] (and is believed to have a larger inapproximability threshold), so this highlights a stark contrast with the MST-interdiction problem.

Our techniques. We give an overview of our algorithm for MST interdiction. Let $\mathrm{val}(R)$ be the w-weight of an MST of $G - R$. Using standard arguments, we can reduce the problem to the following setting (see Section 2 and Theorem 3): we are given interdiction sets $R_1 \subseteq R_2$ with $c(R_1) < B < c(R_2)$ such that $a \cdot \mathrm{val}(R_1) + b \cdot \mathrm{val}(R_2) \geq OPT$, where OPT is the optimal value and $a, b \geq 0$ are such that $a + b = 1$ and $a \cdot c(R_1) + b \cdot c(R_2) = B$. These arguments resemble the ones in [31], but we do not need to assume that the w_e weights are powers of 2. (We emphasize however that this by itself is not the chief source of our improvement.) The technical meat of the algorithm, and where we diverge significantly from [31] to obtain our improved guarantee, is to show how to extract a good interdiction set from R_1, R_2. As mentioned earlier, we replace the greedy algorithm of [31] for extracting a good interdiction set from R_2, and its associated intricate analysis, by considering the tree-knapsack problem to capture the key aspects of this extraction problem, and devise a simple iterative-rounding algorithm that yields a strong LP-relative guarantee for tree knapsack. This conveniently translates to a much-improved 5-approximation algorithm for MST interdiction (Theorem 13). The further improvement to a 4-approximation arises by also leveraging R_1 to find a good interdiction set: instead of focusing solely on R_2 (as done in [31]), we return an interdiction set R such that $R_1 \subseteq R \subseteq R_2$ (see Section 4.1).

To arrive at the tree knapsack problem, observe that $\mathrm{val}(R)$ can be conveniently expressed as a weighted sum of the number of components of $(V, \{e \in E \setminus R : w_e \leq t\})$, where t ranges over some distinct edge weights, say, $0 \leq w_1 < \cdots < w_k$ (Lemma 2). Let $\mathcal{A}_0$ denote the components of $(V, E_{\leq 0} := \emptyset)$, and $\mathcal{A}_i$ denote the components of $(V, E_{\leq i} := \{e \in E \setminus R_2 : w_e \leq w_i\})$ for $i = 1, \ldots, k$. The multiset $\bigcup_{i=0}^{k} \mathcal{A}_i$ forms a laminar family, which can be viewed as a rooted tree. We seek to build our interdiction set R by selecting a suitable collection of sets from this laminar family, ensuring that if we pick a component $A \in \mathcal{A}_i$ then $\delta(A) \cap E_{\leq i}$ is included in R (so that A is a component of $(V, E_{\leq i} \setminus R)$). Whereas $\mathrm{val}(R)$ is nicely *decoupled* across the selected components, it is harder to decouple the interdiction cost incurred and account for it. For instance, summing $c\big(\delta(A) \cap E_{\leq i}\big)$ for the selected $A \in \mathcal{A}_i$ may grossly overestimate the interdiction cost, whereas summing $c\big(\delta(A) \cap \{e : w_e = w_i\}\big)$ for the selected $A \in \mathcal{A}_i$ *underestimates* the interdiction cost. A crucial insight is that, if we ensure that *whenever we pick $A \in \mathcal{A}_i$, we also pick its children in the laminar family*, then *summing $c\big(\delta(A) \cap \{e : w_e = w_i\}\big)$ for the selected $A \in \mathcal{A}_i$ is a good proxy for the interdiction cost incurred.*

This motivates the definition of the tree knapsack problem: given a rooted tree Γ with node values $\{\alpha_v\}$, node weights $\{\beta_v\}$, and budget B, we want to pick a maximum-value *downwards-closed* set of nodes (not containing the root) whose weight is at most B, where downwards-closed means that if we pick a node, then we also pick all its children. The standard knapsack problem is thus the special case where Γ is a star (rooted at its center). We consider the natural LP (TK-P) for tree knapsack, and generalizing a well-known result for knapsack, show that we can efficiently compute a solution of value at least $OPT_{\text{TK-P}} - \max_{\text{chains } C} \sum_{v \in C} \alpha_v$ (Theorem 4), where a chain is a subset of a root-leaf path.

Finally, we show that for the tree-knapsack instance derived (as above) from R_2, $OPT_{\text{TK-P}}$ is "large" (Lemma 10); combining this with the above bound yields our approximation ratio.

Related work. MST interdiction in its full generality seems to have been first considered by [22], who showed that the problem is *NP*-hard. The approximation question for MST interdiction was first investigated by [9]. They focused on the setting with unit interdiction costs, often called the *B-most-vital-edges* problem, showed that this special case remains *NP*-hard, and obtained an $O(\log B)$-approximation (which also yields an $O(\log |E|)$-approximation with general interdiction costs). This guarantee was improved only recently by Zenklusen [31],

who gave the first (and current-best) $O(1)$-approximation algorithm for (general) MST interdiction, achieving an approximation ratio of 14. The B-most-vital edges problem has been well studied for $B = 1$ and for $B = O(1)$, where it can be solved optimally; see, e.g., [21] and the references therein. The special case of MST interdiction where we have only two distinct edge weights captures the *budgeted graph disconnection* (BGD) problem [7] for which a 2-approximation is known [7]. As noted by [31], MST interdiction can be viewed as multilevel-BGD, which makes it much more challenging as it is difficult to control the interactions at the different levels. It is noteworthy that our approximation ratio of 4 for MST interdiction is quite close to the approximation ratio of 2 for BGD.

As with MST interdiction, until recently, there were wide gaps in our understanding of the approximability of the other classic *NP*-hard interdiction problems mentioned earlier. Maximum s-t flow interdiction, even on undirected graphs with unit interdiction costs, is now known to be at least as hard as MinDkS on λ-uniform hypergraphs. This follows from a recent hardness result for *k-route s-t cut* in [13], which turns out to be an equivalent problem.[1] This hardness result has been rediscovered (in a slightly weaker form) by [3], who also gave an $O(n)$-approximation algorithm. For shortest s-t path interdiction, very recently, Lee [20] proved a super-constant hardness result. For maximum-matching interdiction, [6] devised the first $O(1)$-approximation algorithm. Despite this recent progress, interdiction variants of common optimization problems are generally not well understood, especially from the viewpoint of approximability.

The tree knapsack problem was introduced by [15], and is a special case of the *partially-ordered knapsack* (POK) problem [18]. While an FPTAS can be obtained for tree knapsack and some special cases of POK [15, 18], and the natural LP for POK has been investigated [18], our LP-relative guarantee and rounding algorithm for tree knapsack are new.

2 Preliminaries

For any vector $d \in \mathbb{R}^E$ and any subset $F \subseteq E$ of edges, we use $d(F)$ to denote $\sum_{e \in F} d_e$. Given a subset $R \subseteq E$ of edges, we use $\mathrm{val}(R)$, which we call the *value* of R, to denote the w-weight of an MST in the multigraph $G - R$, i.e., $\mathrm{val}(R) := \min_{\text{spanning trees } T \text{ of } G-R} w(T)$. The *minimum-spanning-tree interdiction* problem can thus be restated as follows: $\max \big\{\mathrm{val}(R) : R \subseteq E,\ c(R) \leq B\big\}$.

If there is an interdiction set R with $c(R) \leq B$ such that $G - R$ is disconnected, then $\mathrm{val}(R) = \infty$, and so the MST-interdiction problem is unbounded. Note that this happens iff a min-cut $\delta(S)$ of G satisfies $c\big(\delta(S)\big) \leq B$, and we can efficiently detect this. So in the sequel, we assume that this is not the case. Let OPT denote the optimal value of the MST-interdiction problem (which is now finite). For $F \subseteq E$, let $\sigma(F)$ denote the number of connected components of (V, F).

Let $w_1, w_2, \ldots, w_M$ be the distinct weights in $\{w_e : e \in E\}$, where $0 \leq w_1 < w_2 < \cdots < w_M$. For $i = 1, \ldots, M$, define $E_i := \{e \in E : w_e = w_i\}$ and $E_{\leq i} := \{e \in E : w_e \leq w_i\}$. For notational convenience, we define $w_0 := 0$ and $E_0 = E_{\leq 0} := \emptyset$. (Note that E_0 is *not* necessarily $\{e \in E : w_e = w_0\}$, and $E_{\leq 0}$ is not necessarily $\{e \in E : w_e \leq w_0\}$.)

[1] In k-route s-t cut, the goal is to remove a min w-cost set of edges so as to reduce the s-t edge connectivity to at most $k-1$. This corresponds to taking all but the $k-1$ most-expensive edges of some cut. So we can rephrase this problem as follows: remove at most $k-1$ edges to minimize the (min-s-t-cut value = max-s-t-flow value) with capacities $\{w_e\}$; this is precisely the maximum s-t flow interdiction problem with unit interdiction costs and budget $k-1$.

Let $k \in \{1,\dots,M\}$ be the *smallest* index such that $c\big(\delta(S) \cap E_{\le k}\big) > B$ for every $\emptyset \neq S \subsetneq V$; that is, the multigraph $(V, E_{\le k} \setminus R)$ is connected for all R such that $c(R) \le B$. Note that k is well defined due to our earlier assumption. This implies the following properties, as also observed in [31]:

- **(i)** $OPT \ge w_k$ (since, by definition of k, there is a feasible interdiction set R whose removal disconnects $(V, E_{\le k-1})$);
- **(ii)** for any R with $c(R) \le B$, we have $\mathrm{val}(R) = \mathrm{val}(R \cap E_{\le k-1})$, and hence, there is an optimal solution that only interdicts edges from $E_{\le k-1}$; and
- **(iii)** given (ii), we may add additional edges of weight w_k without impacting the optimal value, so we may assume that (V, E_k) is connected.

We summarize these properties and assumptions below.

▶ **Claim 1.** *Let $k \in \{1,\dots,M\}$ be the smallest index such that $(V, E_{\le k} \setminus R)$ is connected for every $R \subseteq E$ with $c(R) \le B$. Assume that such a k exists. Then, (i) $OPT \ge w_k$, and (ii) there is an optimal solution R^* such that $R^* \subseteq E_{\le k-1}$. Moreover, we may assume that (iii) the multigraph (V, E_k) is connected.*

▶ **Lemma 2.** *Let $R \subseteq E$ be an edge-set such that $(V, E_{\le k} \setminus R)$ is connected. Then* $\mathrm{val}(R) = -w_k + \sum_{i=0}^{k-1} \sigma\left(E_{\le i} \setminus R\right)(w_{i+1} - w_i)$.

Proof. Consider, for example, running Kruskal's algorithm to obtain an MST of $G - R$. We include exactly $\sigma(E_{\le j-1} \setminus R) - \sigma(E_{\le j} \setminus R)$ edges of weight w_j for every $1 \le j \le M$, and this quantity is 0 for all $j > k$. It follows that

$$\begin{aligned}
\mathrm{val}(R) &= \sum_{j=1}^{M} \Big(\sigma\left(E_{\le j-1} \setminus R\right) - \sigma(E_{\le j} \setminus R)\Big) w_j = \sum_{j=1}^{k} \Big(\sigma(E_{\le j-1} \setminus R) - \sigma(E_{\le j} \setminus R)\Big) w_j \\
&= \sum_{j=1}^{k} \Big(\sigma\left(E_{\le j-1} \setminus R\right) - \sigma(E_{\le j} \setminus R)\Big) \sum_{i=0}^{j-1} (w_{i+1} - w_i) \\
&= \sum_{i=0}^{k-1} (w_{i+1} - w_i) \sum_{j=i+1}^{k} \Big(\sigma(E_{\le j-1} \setminus R) - \sigma(E_{\le j} \setminus R)\Big) \\
&= \sum_{i=0}^{k-1} \big(\sigma(E_{\le i} \setminus R) - 1\big)(w_{i+1} - w_i) = -w_k + \sum_{i=0}^{k-1} \sigma(E_{\le i} \setminus R)(w_{i+1} - w_i).
\end{aligned}$$

◀

Given Claim 1, we focus on interdiction sets $R \subseteq E_{\le k-1}$ and recast the MST-interdiction problem as: $\max \big\{\mathrm{val}(R) : R \subseteq E_{\le k-1},\ c(R) \le B\big\}$. As is common in the study of constrained optimization problems (see, e.g., [19, 12] and the references therein), we Lagrangify the budget constraint $c(R) \le B$, and consider the following Lagrangian problem (offset by $-\lambda B$), where $\lambda \ge 0$ is a parameter:

$$\max_{R \subseteq E_{\le k-1}} \quad f_\lambda(R) := \mathrm{val}(R) - \lambda c(R). \tag{P_λ}$$

The expression for $\mathrm{val}(R)$ in Lemma 2 holds for all $R \subseteq E_{\le k-1}$ as (V, E_k) is connected. Since $\sigma(E_{\le i} \setminus R)$ is a supermodular function of R, this implies that $\mathrm{val}(\cdot)$, and hence the objective function $f_\lambda(\cdot)$ of (P_λ), is *supermodular* over the domain $2^{E_{\le k-1}}$: for any $A_1, A_2 \subseteq E_{\le k-1}$, we have $f_\lambda(A_1) + f_\lambda(A_2) \le f_\lambda(A_1 \cap A_2) + f_\lambda(A_1 \cup A_2)$. Hence, (P_λ) can be solved exactly, which we crucially exploit.

Let $\mathcal{O}^*_\lambda$ denote the set of optimal solutions to (P_λ). Observe that for any $\lambda \ge 0$ and any $R \in \mathcal{O}^*_\lambda$, we have $\mathrm{val}(R) - \lambda c(R) \ge OPT - \lambda B$. So if we find some $\lambda \ge 0$ and $R \in \mathcal{O}^*_\lambda$ such

that $c(R) = B$, we have $\text{val}(R) \geq OPT$, so R is an optimal solution. In general, such a pair (λ, R) need not exist, or can be hard to find. However, by doing a binary search for λ, or alternatively, as noted in [31], via parametric submodular-function minimization [8, 23], we can obtain the following result; we include a self-contained proof in the full version.

▶ **Theorem 3** ([31]). *One can find in polytime: either (i) an optimal solution to the MST-interdiction problem, or (ii) a parameter $\lambda \geq 0$ and two optimal solutions R_1, R_2 to (P_λ) such that $R_1 \subseteq R_2$ and $c(R_1) < B < c(R_2)$.*

3 The tree knapsack problem

We now define the *tree knapsack problem*, and devise a simple, clean LP-based approximation algorithm for this problem (Theorem 4, Corollary 6). As we show in Section 4, the tree knapsack problem nicely abstracts the key combinatorial problem encountered in extracting a good interdiction set from an over-budget set R_2 in case (ii) of Theorem 3, and our LP-relative guarantees for tree knapsack readily yield improved approximation guarantees for MST interdiction.

In the tree knapsack problem [15], we have a tree $\Gamma = (\{r\} \cup N, A)$ rooted at node r. Each node $v \in N$ has a *value* $\alpha_v \geq 0$ and a *weight* $\beta_v \geq 0$, and we have a budget B. We say that a subset $S \subseteq N$ of nodes is *downwards-closed* if for every $v \in S$, all children of v are also in S. The goal is to find a maximum-value downwards-closed set $S \subseteq N$ (so $r \notin S$) such that $\sum_{v \in S} \beta_v \leq B$. Observe that the (standard) knapsack problem is precisely the special case of tree knapsack where the underlying tree is a star (rooted at its center). Throughout, we use v to index nodes in N. For $S \subseteq N$ and a vector $\rho \in \mathbb{R}^N$, we use $\rho(S)$ to denote $\sum_{v \in S} \rho_v$.

The following is a natural LP-relaxation for the tree knapsack problem involving variables x_v for all v. Let $\mathsf{ch}(v)$ denote the set of children of node v.

$$\max \quad \sum_v \alpha_v x_v \tag{TK-P}$$

$$\text{s.t.} \quad x_v \leq x_u \qquad \text{for all } v, \text{ for all } u \in \mathsf{ch}(v) \tag{1}$$

$$\sum_v \beta_v x_v \leq B, \qquad 0 \leq x_v \leq 1 \quad \text{for all } v.$$

Tree knapsack was first defined by [15] who devised an FPTAS for this problem via dynamic programming. However, for our purposes, we need an approximation guarantee relative to the above LP, which was not known previously.

The main result of this section is as follows. We say that $C \subseteq N$ is a *chain* if for every two distinct nodes in C, one is a descendant of the other.

▶ **Theorem 4.** *We can compute in polytime an integer solution to* (TK-P) *of value at least $OPT_{\text{TK-P}} - \max_{\text{chains } C \subseteq N} \alpha(C)$.*

Theorem 4 nicely generalizes a well-known result about the standard knapsack problem, namely, that we can always obtain a solution of value at least (LP-optimum)$-\max_v \alpha_v$. Notice that when Γ is a star (i.e., we have a knapsack instance), this is *precisely* the guarantee that we obtain above. The proof of Theorem 4 relies on the following structural result (which extends a similar result known for knapsack). Let $\Gamma(v)$ denote the subtree of Γ rooted at v.

▶ **Lemma 5.** *Let $\bar{x}$ be an extreme-point solution to the linear program* (TK-P). *Then there is at most one child v of r for which the subtree $\Gamma(v)$ contains a fractional node, i.e., some node w with $0 < \bar{x}_w < 1$.*

Proof of Theorem 4. We use *iterative rounding*, and the proof is by induction on the depth d of Γ, which is the maximum number of edges on a root-leaf path.

If $d = 0$, then $N = \emptyset$, and (TK-P) has no variables and constraints, so the statement is vacuously true. So suppose $d \geq 1$. Let x^* be an extreme-point optimal solution of (TK-P). If x^* is integral, then we obtain value $OPT_{\text{TK-P}}$, completing the induction step. Otherwise, by Lemma 5, there is exactly one child v of r such that the subtree $\Gamma(v)$ contains a fractional node.

Set $\tilde{x}' = x^*\big|_{N\setminus\Gamma(v)}$, i.e., x^* restricted to $N\setminus\Gamma(v)$, which is integral. We have $\sum_{u\in N\setminus\Gamma(v)} \alpha_u \tilde{x}'_u = OPT_{\text{TK-P}} - \sum_{w\in\Gamma(v)} \alpha_w x^*_w$. Now consider the tree knapsack instance defined by the tree $\Gamma(v)$ with root v, and budget $B - \sum_{u\in N\setminus\Gamma(v)} \beta_u \tilde{x}'_u$ (and values α_w and weights β_w for all $w \in \Gamma(v) \setminus \{v\}$). Observe that $x^*\big|_{\Gamma(v)\setminus\{v\}}$ is a fractional solution to the LP-relaxation (TK-P) corresponding to this tree knapsack problem, so the optimal value of this LP is at least $\sum_{w\in\Gamma(v)\setminus\{v\}} \alpha_w x^*_w$. (These objects are null if $\Gamma(v) = \{v\}$.) Thus, since $\Gamma(v)$ has depth at most $d - 1$, by our induction hypothesis, our rounding procedure applied to this tree knapsack instance yields an integer solution $\tilde{x}'' \in \{0,1\}^{\Gamma(v)\setminus\{v\}}$ of value at least $\sum_{w\in\Gamma(v)\setminus\{v\}} \alpha_w x^*_w - \max_{\text{chains } C\subseteq\Gamma(v)\setminus\{v\}} \alpha(C)$. Thus, taking $\tilde{x} = (\tilde{x}', \tilde{x}_v = 0, \tilde{x}'')$, we obtain a feasible integer solution to (TK-P) having value at least

$$\begin{aligned} OPT_{\text{TK-P}} - \sum_{w\in\Gamma(v)} \alpha_w x^*_w + \sum_{w\in\Gamma(v)\setminus\{v\}} \alpha_w x^*_w - \max_{\text{chains } C\subseteq\Gamma(v)\setminus\{v\}} \alpha(C) \\ \geq OPT_{\text{TK-P}} - \alpha_v - \max_{\text{chains } C\subseteq\Gamma(v)\setminus\{v\}} \alpha(C) \\ \geq OPT_{\text{TK-P}} - \max_{\text{chains } C\subseteq N} \alpha(C)\ . \end{aligned}$$

This completes the induction step, and hence the proof of the theorem. ◀

We remark that (as is standard) the iterative-rounding procedure in Theorem 4 is in fact combinatorial, since when we move to the subtree $\Gamma(v)$, we only need to move from $x^*\big|_{\Gamma(v)\setminus\{v\}}$ to an extreme-point of the LP of the smaller tree-knapsack instance of no smaller value (instead of obtaining an optimal LP solution), which can be done combinatorially.

We now state a stronger version of Theorem 4 that will be useful in Section 4, where we utilize tree knapsack to solve the MST-interdiction problem. This result follows from a more-careful scrutiny of the proof of Theorem 4. The depth of a node v is the number of edges on the (unique) r-v path of Γ. Let $L_i(\Gamma)$ be the set of nodes of Γ at depth i; we drop Γ if it is clear from the context. For a chain C of Γ, let C_i denote $C \cap L_i(\Gamma)$; note that $|C_i| \leq 1$.

▶ **Corollary 6.** *We can obtain in polytime an integer solution $\tilde{x}$ to* (TK-P) *of value at least* $OPT_{\text{TK-P}} - \max_{\text{chains } C\subseteq N} \{\sum_{i\geq 1:\tilde{x}(L_i)<|L_i|} \alpha(C_i)\}$.

4 MST interdiction

▶ **Theorem 7.** *There is a 4-approximation algorithm for MST interdiction.*

The above theorem is our main technical result. Our guarantee substantially improves the previous-best approximation ratio of 14 obtained by [31]. Also, notably and significantly, our algorithm and analysis, which are based on the tree knapsack problem introduced in Section 3, are noticeably simpler and cleaner than the one in [31]. Improved guarantees for MST interdiction readily follow from (Theorem 4 and) Corollary 6 and Lemma 14, yielding approximation ratios of 5 and 4 respectively for MST interdiction (see Theorem 13 and Section 4.1). The proof below shows a slightly worse guarantee of 5 but introduces the main underlying ideas. Section 4.1 discusses the refinement needed to obtain the 4-approximation.

Our algorithm follows the same high-level outline as the one in [31]. As mentioned earlier, we consider the Lagrangian problem (P_λ), $\max_{R\subseteq E_{\leq k-1}} f_\lambda(R) := \mathrm{val}(R) - \lambda c(R)$, obtained by dualizing the budget constraint $c(R) \leq B$. We then utilize Theorem 3. If this returns an optimal solution, then we are done. So assume in the sequel that Theorem 3 returns $\lambda \geq 0$ and two optimal solutions R_1 and R_2 to (P_λ) such that $R_1 \subseteq R_2$ and $c(R_1) < B < c(R_2)$.

For $R \subseteq E_{\leq k-1}$, define $h(R) := \sum_{i=0}^{k-1} \sigma\left(E_{\leq i} \setminus R\right)(w_{i+1} - w_i) = \mathrm{val}(R) + w_k$. Let $R^* \subseteq E_{\leq k-1}$ denote an optimal solution to the MST-interdiction problem, so $OPT = h(R^*) - w_k$. Let $a, b \geq 0$ such that $a + b = 1$ and $ac(R_1) + bc(R_2) = B$. Then, since $\mathrm{val}(R_1) - \lambda c(R_1) = \mathrm{val}(R_2) - \lambda c(R_2) \geq OPT - \lambda B$, we have $ah(R_1) + bh(R_2) \geq h(R^*)$. We establish our approximation guarantee by comparing the value of our solution against the upper bound $ah(R_1) + bh(R_2) - w_k$. The following claim shows that this upper bound is precisely the optimal value of the Lagrangian relaxation of the MST interdiction problem, which is $UB := \min_{\lambda'\geq 0}\big(\lambda' B + \max_{R\subseteq E_{\leq k-1}} f_{\lambda'}(R)\big)$. Complementing our 4-approximation, we prove a lower bound of 3 on the approximation ratio achievable relative to UB (Section 4.2).

▶ **Claim 8.** *We have $ah(R_1) + bh(R_2) - w_k = UB$.*

Translation to tree knapsack. We now describe how the problem of combining R_1 and R_2 to extract a good, feasible interdiction set can be captured by a suitable instance of the tree knapsack problem defined in Section 3.

For $i = 0, \ldots, k$, let $\mathcal{A}_i \subseteq 2^V$ be the partition of V induced by the connected components of the multigraph $(V, E_{\leq i} \setminus R_2)$. Thus, $\mathcal{A}_k = \{V\}$ and $\mathcal{A}_0 = \{\{v\} : v \in V\}$. The multiset $\bigcup_{i=0}^{k} \mathcal{A}_i$, where we include $S \subseteq V$ multiple times if it lies in multiple $\mathcal{A}_i$s, is a *laminar family* (i.e., any two sets in the collection are either disjoint or one is contained in the other). This laminar family can naturally be viewed as a rooted tree, which defines the tree Γ in the tree knapsack problem. Taking a cue from Lemma 2, we build our interdiction set R by selecting a suitable collection of sets from this laminar family, ensuring that if we pick some $A \in \mathcal{A}_i$, then we include all edges of $\delta(A) \cap E_{\leq i}$ in R and create A as a component of $(V, E_{\leq i} \setminus R)$ (and hence contribute $w_{i+1} - w_i$ to $h(R)$). Formally, the tree Γ has a node $v^{A,i}$ for every component $A \in \mathcal{A}_i$ and all $i = 0, \ldots, k$. For $i > 0$, the children of $v^{A,i}$ are the nodes $\{v^{S,i-1} : S \in \mathcal{A}_{i-1}, S \subseteq A\}$. Thus, Γ has depth k and root $r = v^{V,k}$. Recall that $L_i := L_i(\Gamma)$ denotes the set of nodes of Γ at depth i, which correspond to the components in $\mathcal{A}_{k-i}$ here. Let N be the set of non-root nodes of Γ.

For a node $v^{A,i} \in N$ (so $0 \leq i < k$), define its value $\alpha_{v^{A,i}} := w_{i+1} - w_i$. Let $R(v^{A,i}) := \delta(A) \cap E_i$ (which is $\emptyset$ for every leaf $v^{A,0}$). Define the weight of $v^{A,i}$ to be $\beta_{v^{A,i}} := c\big(R(v^{A,i})\big)$. For $N' \subseteq N$, let $R(N') := \bigcup_{q\in N'} R(q)$. Observe that $R(N) \subseteq R_2$. We set the budget of the tree-knapsack instance to B, the budget for MST interdiction.

The intuition is that we want to encode that picking node $v^{A,i}$ corresponds to creating component A in the multigraph $(V, E_{\leq i} \setminus R)$, where R is our interdiction set, in which case $\alpha_{v^{A,i}}$ gives the contribution from A to $h(R)$. However, in order to pay for the interdiction cost $c\big(\delta(A)\big)$ incurred, we need to take the β_q weights of all nodes q in the subtree rooted at $v^{A,i}$. Therefore, we insist that if we pick $v^{A,i}$ then we pick all its descendants (i.e., we pick a downwards-closed set of nodes), and then $\sum_{q\in\Gamma(v^{A,i})} \alpha_q$ gives the contribution from the components created to $h(R)$. Lemma 9 formalizes this intuition, and shows that if $N' \subseteq N$ is a downwards-closed set of nodes, then $\beta(N')$ and $\alpha(N')$ are good proxies (roughly speaking) for the interdiction cost $c\big(R(N')\big)$ incurred and $h\big(R(N')\big)$ respectively.

▶ **Lemma 9.** *Let $N' \subseteq N$ be downwards closed, and $R = R(N')$. Then*
(i) *$\beta(N')/2 \leq c(R) \leq \beta(N')$; and*
(ii) *$h(R) = \mathrm{val}(R) + w_k \geq \alpha(N') + \sum_{0\leq i\leq k-1 : L_{k-i}\setminus N' \neq \emptyset} (w_{i+i} - w_i)$.*

Proof. Each edge in R appears in at least one, and at most two, of the sets $\{R(q)\}_{q\in N'}$, so $\frac{1}{2}\sum_{q\in N'} c\big(R(q)\big) \le c(R) \le \sum_{q\in N'} c\big(R(q)\big)$. This yields part (i) since $\beta(N') = \sum_{q\in N'} c\big(R(q)\big)$.

For part (ii), consider an index $0 \le i \le k-1$. Since N' is downwards closed, for every node $v^{A,i} \in N'$, all descendants of $v^{A,i}$ are in N'; so $R \supseteq \delta(A) \cap E_{\le i}$ and A is a connected component of $(V, E_{\le i} \setminus R)$. Further, note that if $L_{k-i} \setminus N' \neq \emptyset$, then the sets $\{A : v^{A,i} \in N'\}$ do not cover V entirely, and so $(V, E_{\le i} \setminus R)$ must have at least one additional connected component. It follows that $(V, E_{\le i} \setminus R)$ always has at least $\min\big\{|N' \cap L_{k-i}| + 1, |L_{k-i}|\big\}$ connected components. Plugging this in Lemma 2 yields the result. ◀

▶ **Lemma 10.** *The vector* $\hat{x} := \big(\hat{x}_q = \frac{b}{2}\big)_{q\in N}$ *is a feasible solution to* (TK-P) *for the above tree-knapsack instance* $(\Gamma, \{\alpha_q\}, \{\beta_q\}, B)$. *Hence,* $OPT_{TK\text{-}P} \ge \frac{b}{2} \cdot h(R_2)$.

Proof. It is clear that $\hat{x}$ satisfies (1), and $0 \le \hat{x}_q \le 1$ for all $q \in N$. Applying Lemma 9 to $N' = N$ (which is indeed downwards-closed), we obtain $\beta(N) \le 2c\big(R(N)\big) \le 2c(R_2)$. So $\sum_{q\in N} \beta_q \hat{x}_q \le b \cdot c(R_2) \le a \cdot c(R_1) + b \cdot c(R_2) = B$. Finally, $OPT_{\text{TK-P}}$ is at least the objective value of $\hat{x}$, which is $\frac{b}{2} \cdot \alpha(N) = \frac{b}{2} \cdot h(R_2)$. ◀

Given this translation between tree knapsack and MST interdiction, it is easy to see that Corollary 6 (coupled with Lemmas 9 and 10) yields the following guarantee, which directly leads to an improved approximation guarantee of 5 for MST interdiction (see Claim 12).

▶ **Lemma 11** (Consequence of Corollary 6, Lemmas 9 and 10)**.** *We can obtain a feasible interdiction set* R *such that* $h(R) \ge \frac{b}{2} \cdot h(R_2)$.

▶ **Claim 12.** *We have* $\max\big\{w_k, h(R_1) - w_k, \frac{b}{2} \cdot h(R_2) - w_k\big\} \ge UB/5 \ge OPT/5$.

Proof. We have

$$\begin{aligned}\max\Big\{w_k, h(R_1) - w_k, \tfrac{b}{2} \cdot h(R_2) - w_k\Big\} &\ge \tfrac{2-b}{5-2b} \cdot w_k + \tfrac{1-b}{5-2b} \cdot \big(h(R_1) - w_k\big) \\ &\quad + \tfrac{2}{5-2b} \cdot \big(\tfrac{b}{2} \cdot h(R_2) - w_k\big) \\ &= \frac{1}{5-2b}\Big(ah(R_1) + bh(R_2) - w_k\Big) = \frac{UB}{5-2b} \ge UB/5 \ge OPT/5.\end{aligned}$$ ◀

▶ **Theorem 13.** *There is a* 5*-approximation algorithm for MST interdiction.*

Proof. If Theorem 3 returns an optimal solution, we are done. Otherwise, we return the best among a min-cut of $(V, E_{\le k-1})$, the set R_1, and the interdiction set returned by Lemma 11. The proof now follows from Claim 12. ◀

4.1 Improvement to the guarantee stated in Theorem 7

The improved approximation guarantee of 4 comes from the fact that instead of focusing only on R_2, we now *interpolate* between R_1 and R_2 to obtain our interdiction set R, i.e., we return R such that $R_1 \subseteq R \subseteq R_2$. Since we always include R_1, we change the definition of the tree-knapsack instance that we create accordingly. The tree Γ and the node weights $\{\alpha_q\}$ are unchanged; the weight of $v^{A,i}$ is now $\beta^{\text{new}}_{v^{A,i}} := c\big(R^{\text{new}}(v^{A,i})\big)$, where $R^{\text{new}}(v^{A,i}) := R(v^{A,i}) \setminus R_1 = \big(\delta(A) \setminus R_1\big) \cap E_i$, and our budget is $B^{\text{new}} := B - c(R_1)$. For $N' \subseteq N$, define $R^{\text{new}}(N') := R_1 \cup \bigcup_{q\in N'} R^{\text{new}}(q)$. Observe that $R^{\text{new}}(N) \subseteq R_2$.

Since $R_1 \subseteq R_2$, each component U of $(V, E_{\le i} \setminus R_1)$ is a union of components of $(V, E_{\le i} \setminus R_2)$, and hence, maps to a subset S of the nodes of Γ at depth $k-i$. We exploit the fact that since we include R_1 in our interdiction set, if we pick ℓ nodes from S, then we create $\min\{\ell+1, |S|\}$

components within U; this $+1$ term that we accrue (roughly speaking) from all components of $(V, E_{\le j} \setminus R)$ over all $j = 0, \ldots, k-1$ is the source of our improvement.

The following variant of Corollary 6 exploits the structure of the tree-knapsack instance obtained from the MST-interdiction problem, which we then utilize to obtain an interdiction set with an improved bound on $h(R)$ (Lemma 15).

▶ **Lemma 14.** *Let $\big(\Gamma, \{\alpha_v\}, \{\beta_v\}, B\big)$ be an instance of the tree knapsack problem such that $\alpha_v = \alpha^{(i)}$ for all $v \in L_i(\Gamma)$ and all $i \ge 1$. Let $\mathcal{S}_i$ be a partition of $L_i(\Gamma)$ for all $i \ge 1$. Let $\theta \in [0,1]$ be such that $(\hat{x}_q = \theta)_{q \in N}$ is a feasible solution to* (TK-P)*. We can obtain in polytime an integer solution $\tilde{x}$ to* (TK-P) *such that*

$$\sum_{i \ge 1} \sum_{S \in \mathcal{S}_i} \alpha^{(i)} \min\{\tilde{x}(S) + 1, |S|\} \ge \sum_{i \ge 1} \alpha^{(i)} |L_i| \theta + \sum_{i \ge 1 : |\mathcal{S}_i| > 1} \alpha^{(i)} \Big((1-\theta)|\mathcal{S}_i| - 1 \Big).$$

▶ **Lemma 15.** *Using Lemma 14, we can obtain a feasible interdiction set R such that $h(R) \ge \frac{a}{2} \cdot h(R_1) + \frac{b}{2} \cdot h(R_2) - \frac{a}{2} \cdot w_k$.*

Proof. For $i = 0, \ldots, k$, let $\mathcal{B}_i$ denote the partition of V induced by the connected components of $(V, E_{\le i} \setminus R_1)$. Since $R_1 \subseteq R_2$, the partition $\mathcal{A}_i$ refines (not necessarily strictly) the partition $\mathcal{B}_i$ for all $i = 0, \ldots, k$. The components in $\mathcal{B}_{k-i}$ therefore naturally induce a partition $\mathcal{S}_i$ of the nodes of Γ at depth i, consisting of the sets $\{v^{A,k-i} : A \in \mathcal{A}_{k-i}, A \subseteq S\}_{S \in \mathcal{B}_{k-i}}$.

We apply Lemma 14 to the tree-knapsack instance $\big(\Gamma, \{\alpha_q\}, \{\beta_q^{\mathsf{new}}\}, B^{\mathsf{new}}\big)$, taking $\alpha^{(i)} = w_{k-i+1} - w_{k-i}$ and $\mathcal{S}_i$ to be the partition defined above, for all $i = 1, \ldots, k$, and $\theta = \frac{b}{2}$. We show that $\hat{x} := (\hat{x}_q = \theta)_{q \in N}$ is a feasible solution to (TK-P) for this tree-knapsack instance. This follows because $\beta^{\mathsf{new}}(N) = 2c\big(\bigcup_{q \in N} R^{\mathsf{new}}(q)\big) \le 2\big(c(R_2) - c(R_1)\big)$ and $(1-b) \cdot c(R_1) + b \cdot c(R_2) = B$, so we have $\sum_q \beta_q^{\mathsf{new}} \hat{x}_q = \theta \beta^{\mathsf{new}}(N) \le b\big(c(R_2) - c(R_1)\big) = B - c(R_1) = B^{\mathsf{new}}$.

Let $\tilde{x}$ be the integer solution returned by Lemma 14, which specifies a downwards-closed set $N' \subseteq N$. Let $R = R^{\mathsf{new}}(N')$. We first show that, analogous to Lemma 9, R is feasible, and $h(R) \ge g(\tilde{x}) := \sum_{i \ge 1} \sum_{S \in \mathcal{S}_i} \alpha^{(i)} \min\{\tilde{x}(S) + 1, |S|\}$. We have

$$\begin{aligned} c(R) = c(R_1) + c\Big(\bigcup_{q \in N'} R^{\mathsf{new}}(q) \Big) &\le c(R_1) + \sum_{q \in N'} c\big(R^{\mathsf{new}}(q)\big) \\ &= c(R_1) + \beta^{\mathsf{new}}(N') \le c(R_1) + B^{\mathsf{new}} = B. \end{aligned}$$

Consider any index $0 \le i \le k-1$. As in the proof of part (ii) of Lemma 9, for every node $v^{A,i} \in N'$, we know that A is a component of $(V, E_{\le i} \setminus R)$. Consider any $S \in \mathcal{S}_{k-i}$, and let $U = \bigcup_{v^{A,i} \in S} A$. Note that if $S \setminus N' \ne \emptyset$, then $\bigcup_{v^{A,i} \in S \setminus N'} A$ is non-empty. So there are always at least $\min\big\{|N' \cap S| + 1, |S|\big\}$ components of $(V, E_{\le i} \setminus R)$ contained in U. Therefore, by Lemma 2 (and since $\mathcal{S}_i$ is a partition of L_i for each i), we obtain

$$h(R) = \mathrm{val}(R) + w_k \ge \sum_{i=0}^{k-1} \sum_{S \in \mathcal{S}_{k-i}} (w_{i+1} - w_i) \min\big\{|N' \cap S| + 1, |S|\big\} = g(\tilde{x}).$$

The guarantee in Lemma 14 then yields the following. Recall that $a = 1 - b$.

$$\begin{aligned} h(R) &\ge \sum_{i=0}^{k-1} (w_{i+1} - w_i) \sigma(E_{\le i} \setminus R_2) \cdot \tfrac{b}{2} + \sum_{\substack{i=0,\ldots,k-1: \\ \sigma(E_{\le i} \setminus R_1) > 1}} (w_{i+1} - w_i) \Big[\Big(1 - \tfrac{b}{2}\Big) \sigma(E_{\le i} \setminus R_1) - 1 \Big] \\ &\ge \tfrac{b}{2} \cdot h(R_2) + \sum_{\substack{i=0,\ldots,k-1: \\ \sigma(E_{\le i} \setminus R_1) > 1}} (w_{i+1} - w_i) \sigma(E_{\le i} \setminus R_1) \cdot \tfrac{a}{2} \end{aligned} \tag{2}$$

where inequality (2) follows since $t\big(1-\frac{b}{2}\big)-1 \geq t(1-b)/2$ for all $t \geq 2$. The RHS of (2) is

$$\tfrac{b}{2}\cdot h(R_2)+\tfrac{a}{2}\cdot h(R_1)-\sum_{\substack{i=0,\dots,k-1:\\ \sigma(E_{\leq i}\setminus R_1)=1}}(w_{i+1}-w_i)\cdot\tfrac{a}{2}\geq \tfrac{a}{2}\cdot h(R_1)+\tfrac{b}{2}\cdot h(R_2)-\tfrac{a}{2}\cdot w_k. \quad \blacktriangleleft$$

Proof of Theorem 7. We either return an optimal solution found by Theorem 3, or return the better of a min-cut of $(V, E_{\leq k-1})$ and the interdiction set returned by Lemma 15. We obtain a solution of value $\max\big\{w_k, \frac{a}{2}\cdot h(R_1)+\frac{b}{2}\cdot h(R_2)-\big(1+\frac{a}{2}\big)w_k\big\}$, which is at least

$$\frac{1+a}{3+a}\cdot w_k+\frac{2}{3+a}\cdot\Big(\tfrac{a}{2}\cdot h(R_1)+\tfrac{b}{2}\cdot h(R_2)-\big(1+\tfrac{a}{2}\big)w_k\Big)=\frac{UB}{3+a}\geq OPT/4. \quad \blacktriangleleft$$

4.2 Lower bound on the approximation ratio achievable relative to UB

We show that for every $\epsilon > 0$, there exist MST-interdiction instances, where $UB/OPT \geq 3-\epsilon$. This implies that one cannot achieve an approximation ratio better than 3 when comparing against the upper bound UB used in our analysis (and the one in [31]).

▶ **Theorem 16.** *For any $\epsilon > 0$, there exists an MST-interdiction instance with $\frac{UB}{OPT} \geq 3-\epsilon$.*

Proof. Our instance is a graph $G=(V,E)$, where $V := \{v_1,\dots,v_n\}$ with $n \geq \min\{4, 4/\epsilon\}$. The edge set is $E = E_1 \cup E_2$, where $E_1 := \{v_1v_2, v_2v_3, \dots, v_{n-2}v_{n-1}, v_{n-1}v_1\}$ is a simple cycle on $v_1,\dots,v_{n-1}$, and $E_2 := \{v_1v_n, v_2v_n, \dots, v_{n-1}v_n\}$ is a star rooted at v_n with leaves $v_1,\dots,v_{n-1}$. The edges in E_1 have weight $w_1 = 0$ and interdiction cost n, while the edges in E_2 have weight $w_2 = 1$ and interdiction cost $2n$. The interdiction budget is $B = 2n-2$.

Observe that the index k defined in Claim 1 is equal to 2. This also implies that $\mathrm{val}(R) \leq 1$ for any feasible interdiction set R: since $R \subseteq E_1$ and $|R \cap E_1| \leq 1$, we can construct a spanning tree of $G-R$ by taking $n-2$ edges from $E_1 \setminus R$ and any edge from E_2. So $OPT = 1$.

Now we proceed to compute the upper bound UB. For $R \subseteq E_{\leq 1}$, we have $f_\lambda(R) = 1$ if $R = \emptyset$, and $|R|(1-n\lambda)$ otherwise. Therefore, $\eta(\lambda) := \lambda B + \max_{R\subseteq E_{\leq 1}} f_\lambda(R) = \lambda B + \max\big\{1, (n-1)(1-n\lambda)\big\} = \max\big\{\lambda B+1, (n-1)-\lambda\big(n(n-1)-B\big)\big\}$, which is minimized at $\lambda = \frac{n-2}{n(n-1)}$. Therefore $UB := \min_{\lambda\geq 0}\eta(\lambda) = \frac{2(n-2)}{n}+1 = 3-\frac{4}{n} \geq (3-\epsilon)OPT$. $\blacktriangleleft$

5 Extension to metric-TSP interdiction

In the *metric-TSP interdiction* problem, we are given a complete graph $G=(V,E)$ with metric edge weights $\{w_e\}_{e\in E}$ and nonnegative interdiction costs $\{c_e\}_{e\in E}$, along with a nonnegative budget B. The goal is to find a set of edges $R \subseteq E$ such that $c(R) \leq B$ so as to maximize the minimum w-weight of a closed walk in the graph $G-R$ that visits each vertex at least once. Zenklusen [31] observed that an α-approximation algorithm for the MST interdiction problem yields a 2α-approximation algorithm for the metric-TSP interdiction problem. As a corollary to our Theorem 7, we therefore obtain the following result.

▶ **Theorem 17.** *There is an 8-approximation algorithm for metric-TSP interdiction.*

6 Maximum-spanning-tree interdiction

We now consider the *maximum-spanning-tree* (MaxST) *interdiction* problem, wherein the input $\big(G=(V,E), \{w_e \geq 0\}_{e\in E}, \{c_e \geq 0\}_{e\in E}, B\big)$ is the same as in the MST interdiction problem, but the goal is to remove a set $R \subseteq E$ of edges with $c(R) \leq B$ so as to *minimize* the

w-weight of a *maximum spanning tree* of $G-R$. We show that this problem is at least as hard as the minimization version of the *densest-k-subgraph* problem (MinDkS), wherein we seek a minimum-size set S of nodes in a given graph such that at least k edges have both endpoints in S. This shows a stark contrast between MST interdiction and MaxST interdiction.

▶ **Theorem 18.** *An $\alpha(m,n)$-approximation algorithm for the maximum-spanning-tree interdiction problem for instances with m edges, n nodes, yields a $2\alpha(m+n-1,n)$-approximation algorithm for* MinDkS *for instances with m edges and n nodes.*

Proof. Let $\mathcal{I} = \big(H=(N,F),k\big)$ be a MinDkS instance, with $|N|=n$, $|F|=m$. We may assume that $|F| \geq k$ as otherwise the instance is infeasible. We construct the following MaxST-interdiction instance $\mathcal{I}'$. The multigraph is $G=(N, E := E' \cup F)$, where E' is an arbitrary tree spanning N. Set $w_e = 0$, $c_e = m-k+1$ for all $e \in E'$, and $w_e = c_e = 1$ for all $e \in F$. We set the budget to $B = m-k$. Thus, if $R \subseteq E$ satisfies $c(R) \leq B$, we must have $R \subseteq F$, and so $G-R$ is connected and the interdiction problem has a finite optimal value.

We show that: (1) if $R \subseteq F$ is a feasible interdiction set, then the set S of non-isolated nodes of $(N, F \setminus R)$ is a feasible MinDkS solution of value at most $2 \cdot \mathsf{MaxST}(G-R)$, where $\mathsf{MaxST}(G-R)$ is the weight of a maximum spanning tree of $G-R$; (2) conversely, if $S \subseteq N$ is a feasible MinDkS solution, then $F \setminus F(S)$ is a feasible interdiction set with objective value at most $|S|$, where $F(S)$ is the set of edges in F having both endpoints in S.

These two statements imply the theorem as follows. Let $\mathcal{A}$ be the stated $\alpha = \alpha(m+n-1,n)$-approximation algorithm for maximum-spanning-tree interdiction. We run $\mathcal{A}$ to obtain a feasible interdiction set R, which yields a corresponding MinDkS solution S. Then,

$$|S| \leq 2 \cdot \mathsf{MaxST}(G-R) \leq 2\alpha\, OPT(\mathcal{I}') \leq 2\alpha\, OPT(\mathcal{I}) \ ,$$

where the first and last inequalities follow from statements (1) and (2) above.

We now prove statements (1) and (2). Let $R \subseteq F$ be such that $c(R) = |R| \leq B$. Let S denote the set of non-isolated vertices in the graph $(N, F \setminus R)$, so every node in S has at least one edge of $F \setminus R$ incident to it. First, we argue that S is a feasible MinDkS-solution. Since each vertex of $N \setminus S$ is isolated in the graph $(N, F \setminus R)$, it follows that $R \supseteq F \setminus F(S)$. Therefore, $|F| - |F(S)| \leq |R| \leq B = m-k$, and so $|F(S)| \geq k$. The weight of a maximum spanning tree in $G-R$ is equal to $|S| - \sigma$, where σ is the number of connected components of the graph $(S, F \setminus R)$. By the definition of S, this multigraph has no isolated vertices. So $\sigma \leq |S|/2$, and therefore $\mathsf{MaxST}(G-R) = |S| - \sigma \geq |S|/2$. This proves (1).

Conversely, suppose $S \subseteq N$ is such that $|F(S)| \geq k$. Then $R = F \setminus F(S)$ satisfies $c(R) = m - |F(S)| \leq B$, so is a feasible interdiction set. We have $\mathsf{MaxST}(G-R) = |S| - \sigma \leq |S|$, where σ is the number of connected components of $(S, F \setminus R)$. This proves (2). ◀

The above hardness result continues to hold with unit interdiction costs, since we can replace each edge e with $c_e = m-k+1$ in the above reduction with $m-k+1$ parallel unit-cost edges (of weight 0). Our reduction creates a MaxST-interdiction instance with two distinct edge weights $w_1 < w_2$. This interdiction problem can be seen as a special case of the following *matroid interdiction* problem (involving the graphic matroid on $\{e \in E : w_e = w_2\}$): given a matroid with ground set U and rank function rk, interdiction costs $c : U \mapsto \mathbb{R}_+$, and budget B, minimize $\mathrm{rk}(U \setminus R)$ subject to $c(R) \leq B$. Our hardness result for MaxST interdiction thus also implies that matroid interdiction is MinDkS-hard. A related rank-reduction problem—minimize $c(R)$ subject to $\mathrm{rk}(U \setminus R) \leq \mathrm{rk}(U) - k$—was considered by [16] and shown to be MinDkS-hard for transversal matroids (but not for graphic matroids, wherein this is essentially the min k-cut problem).

We remark that it is possible to achieve *bicriteria* approximation guarantees for MaxST interdiction: we can obtain a solution of weight $W \leq (1+\epsilon)OPT$ while violating the budget by a $\left(1+\frac{1}{\epsilon}\right)$ factor (and $W > OPT$ implies no budget violation). This follows by taking $\lambda = \epsilon OPT/B$ in the Lagrangian problem $\min_R\big(\mathsf{MaxST}(G-R)+\lambda c(R)\big)$, which is a submodular minimization problem that can be solved exactly; it also follows from the work of [4].

References

1. N. Assimakopoulos. A network interdiction model for hospital infection control. *Computers in Biology and Medicine*, 17(6):413–422, 1987.
2. C. Burch, R. Carr, S. Krumke, M. Marathe, C. Phillips, and E. Sundberg. A decomposition-based pseudoapproximation algorithm for network flow inhibition. In *Network Interdiction and Stochastic Integer Programming*, chapter 3, pages 51–68. Springer, 2003.
3. S. Chestnut and R. Zenklusen. Hardness and approximation for network flow interdiction. *CS arXiv*, November 2015.
4. S. Chestnut and R. Zenklusen. Interdicting structured combinatorial optimization problems with $\{0,1\}$-objectives. *CS arXiv*, November 2015.
5. R. L. Church, M. P. Scaparra, and R. S. Middleton. Identifying critical infrastructure: the median and covering facility interdiction problems. *Annals of the Association of American Geographers*, 94(3):491–502, 2004.
6. M. Dinitz and A. Gupta. Packing interdiction and partial covering problems. In *Proceedings of 16th IPCO*, pages 157–168, 2013.
7. A. Engelberg, J. Könemann, S. Leonardi, and J. Naor. Cut problems in graphs with a budget constraint. *Journal of Discrete Algorithms*, 5:262–279, 2007.
8. L. Fleischer and S. Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 131(2):311–322, 2003.
9. G. N. Frederickson and R. Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33:244–266, 1999.
10. D. R. Fulkerson and G. C. Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Math. Programming*, 13:116–118, 1977.
11. P. M. Ghare, D. C. Montgomery, and W. C. Turner. Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly*, 18:37–45, 1971.
12. F. Grandoni, R. Ravi, M. Singh, and R. Zenklusen. New approaches to multi-objective optimization. *Mathematical Programming*, 146(1-2):525–554, 2014.
13. G. Guruganesh, L. Sanità, and C. Swamy. Improved region-growing and combinatorial algorithms for k-route cut problems. In *Proceedings of SODA*, pages 676–695, 2015.
14. E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.
15. D. S. Johnson and K. A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Math. of Oper. Research*, 8(1):1–14, 1983.
16. G. Joret and A. Vetta. Reducing the rank of a matroid. *Discrete Mathematics & Theoretical Computer Science*, 17(2):143–156, 2015.
17. L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: total and node-wise limited interdiction. *Theoretical Computer Science*, 43(2):204–233, 2008.
18. S. Kolliopoulos and G. Steiner. Partially-ordered knapsack and applications to scheduling. *Discrete Applied Mathematics*, 155(8):889–897, 2007.
19. J. Könemann, O. Parekh, and D. Segev. A unified approach to approximating partial covering problems. *Algorithmica*, 59(4):489–509, 2011.

20 Euiwoong Lee. Improved hardness for cut, interdiction, and firefighter problems. *CS arXiv*, July 2016.

21 W. Liang. Finding the k most vital edges with respect to minimum spanning trees for fixed k. *Discrete Applied Mathematics*, 113(2-3):319–327, 2001.

22 K. Liri and M. Chern. The most vital edges in the minimum spanning tree problem. *Information Processing Letters*, 45:25–31, 1993.

23 K. Nagano. A faster parametric submodular function minimization algorithm and applications. Technical report, University of Tokyo, 2007. METR 2007-43.

24 F. Pan, W. Charlton, and D. P. Morton. Stochastic network interdiction of nuclear material smuggling. In D.L. Woodruff, editor, *Network Interdiction and Stochastic Integer Programming*, pages 1–19. Kluwer Academic Publishers, 2002.

25 C. A. Phillips. The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 776–785, 1993.

26 P. Raghavendra and D. Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the 42nd STOC*, pages 755–764, 2010.

27 J. Salmeron, K. Wood, and R. Baldick. Worst-case interdiction analysis of large-scale electric power grids. *IEEE Trans. on Power Systems*, 24(1):96–104, 2009.

28 R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modeling*, 17(2):1–18, 1993.

29 R. Zenklusen. Matching interdiction. *Discrete App. Math.*, 158:1676–1690, 2010.

30 R. Zenklusen. Network flow interdiction on planar graphs. *Discrete Applied Mathematics*, 158(13):1441–1455, 2010.

31 R. Zenklusen. An $O(1)$-approximation for minimum spanning tree interdiction. In *Proceedings of the 56th FOCS*, pages 709–728, 2015.

Reordering Buffer Management with a Logarithmic Guarantee in General Metric Spaces

Matthias Kohler[1] **and Harald Räcke**[2]

1 **Department of Informatics, Technical University of Munich, Munich, Germany**
`kohler@in.tum.de`
2 **Department of Informatics, Technical University of Munich, Munich, Germany**
`raecke@in.tum.de`

Abstract

In the reordering buffer management problem a sequence of requests arrive online in a finite metric space, and have to be processed by a single server. This server is equipped with a request buffer of size k and can decide at each point in time, which request from its buffer to serve next. Servicing of a request is simply done by moving the server to the location of the request. The goal is to process all requests while minimizing the total distance that the server is travelling inside the metric space.

In this paper we present a deterministic algorithm for the reordering buffer management problem that achieves a competitive ratio of $O(\log \Delta + \min\{\log n, \log k\})$ in a finite metric space of n points and aspect ratio Δ. This is the first algorithm that works for general metric spaces and has just a logarithmic dependency on the relevant parameters. The guarantee is memory-robust, i.e., the competitive ratio decreases only slightly when the buffer-size of the optimum is increased to $h = (1 + \epsilon)k$. For memory robust guarantees our bounds are close to optimal.

1998 ACM Subject Classification F.1.2 [Modes of Computation] Online Computation

Keywords and phrases Online algorithms, reordering buffer, metric spaces, scheduling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.33

1 Introduction

In the reordering buffer management problem a sequence of requests arrive online in a finite metric space, and have to be processed by a single server. This server is equipped with a request buffer and can decide at each point in time, which request from its buffer to serve next. Servicing of a request is simply done by moving the server to the location of the request. The goal is to process all requests while minimizing the total distance that the server is traveling inside the metric space.

This simple, abstract model can be used for modeling context switching costs that occur in various applications in many different areas ranging from production engineering through computer graphics to information retrieval [7, 10, 17, 21]. In the online version of the problem the server does not see future requests but has to make its decision based on past requests and the requests it currently holds in the request buffer. The worst case ratio between the cost of the online algorithm and the cost of an optimal offline algorithm is called the *competitive ratio*.

We say a guarantee on the competitive ratio for the reordering buffer management problem is *memory robust* if the guarantee degrades gracefully as the buffer-size of the optimum algorithm is increased over the buffer-size of the online algorithm. More precisely, the ratio

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 33; pp. 33:1–33:12

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

between the cost of the online algorithm with a buffer-size of k, and the cost of an optimum offline algorithm with a buffer-size of $h \geq k$ should be at most $O(ch/k)$.

The main result of this paper is a deterministic algorithm for the reordering buffer management problem that achieves a competitive ratio of $O(h(\log \Delta + \min\{\log n, \log k\})/k)$ in a finite metric space of n points and aspect ratio Δ. The algorithm is also $O(h)$-competitive because any reasonable algorithm achieves this competitive ratio (see Lemma 13 in the appendix). This is the first algorithm that works for general metric spaces and has just a logarithmic dependency on the relevant parameters. The algorithm and its analysis are also very simple.

It has been shown that even on a uniform metric ($\Delta = 1$) the competitive ratio of an online algorithm (deterministic or randomized) must be $\Omega(\log k)$ against an optimum algorithm with buffer-size $h \geq (1+\epsilon)k$ [1, 12]. Hence, an $O(\log k)$ term in the competitive ratio is unavoidable for memory robust algorithms.

Biénkowski et al. [9] have shown that for a sub-linear dependency on the buffer-size there needs to be another term in the competitive ratio apart from k for memory-robust algorithms. In particular, they give an instance on a line metric with n equidistant points for which any online algorithm looses a factor of $\Omega(\min\{k, \log n\})$ against an optimum algorithm with slightly larger buffer ($h = (1+\epsilon)k$). For this instance the number of points n is equal to the aspect ratio Δ. So a logarithmic dependency on the aspect ratio seems reasonable.

Englert and Räcke [12] present a deterministic, memory-robust algorithm for tree metrics of hop-diameter D that obtains a competitive ratio of $O(\frac{h}{k}(\log D + \log k))$. They then use the technique of approximating arbitrary metrics by tree-metrics due to Fakcharoenphol, Rao, and Talwar [14] to obtain a randomized $O(\frac{h}{k} \log n \cdot \log h)$-competitive algorithm for general metrics.

As a whole these results are uncomparable to our results. There exist metrics where the aspect ratio Δ is very small, but there are a lot of points, resulting in very poor guarantees from the result by Englert and Räcke. However, if one considers a star with edges of different length, the aspect ratio could be very high, but the hop-diameter in the tree is just 2, which makes the guarantee given by the result in [12] stronger than ours. One advantage of our result is that for general metrics the dependency on our parameters is logarithmic, while Englert and Räcke have the product of two logarithms. This product cannot easily be removed as any (memory-robust) algorithm that relies on the FRT-approximation will loose one logarithm because of FRT, and another because an online solution on a tree will have a logarithmic competitive ratio.

In Section 5 we deal with the question whether it is possible to trade the dependency on $\log \Delta$ in our competitive ratio for something else, like e.g. $\log n$, which does not depend on the aspect ratio. Our algorithm is memory restricted in the sense that it makes its decisions only depending on the content of the buffer, and on the content of an additional memory that contains k bits. We show, that for such a scenario there exist instances with an aspect ratio Δ, on which every memory-restricted algorithm with k bits is $\Omega(\sqrt{\log \Delta})$-competitive. This extends a lower bound due to Khandekar and Pandit [20] for memoryless algorithms.

1.1 Further Related Work

Most previous work on the reordering buffer management problem considers the case of uniform metrics. Räcke et al. [22] introduced the problem and developed a deterministic algorithm with competitive ratio $O(\log^2 k)$, which was subsequently improved to $O(\log k)$ by Englert and Westermann [13]. The analysis of both these algorithms can be slightly modified to give a memory-robust guarantee.

The first paper that used an analysis technique that is not memory-robust (and can therefore beat the $\Theta(\log k)$-guarantee) was due to Avigdor-Elgrabli and Rabani, who presented a deterministic algorithm with competitive ratio $O(\log k/\log\log k)$. This in turn was improved to a guarantee of $O(\sqrt{\log k})$ by Adamaszek et al. [2], which is close to optimal due to a lower bound of $\Omega(\sqrt{\log k/\log\log k})$ shown in the same paper. For randomized algorithms Avigdor-Elgrabli and Rabani present an $O(\log\log k)$-competitive algorithm [6]. This is optimal due to a corresponding lower bound proved by Adamaszek et al. [2].

For star metric spaces the result by Englert and Westermann [13] obtains a deterministic competitive ratio of $O(\log k)$. The result by Adamaszek et al. [2] gives an $O(\sqrt{\log k})$ guarantee for the case that $\Delta = O(\text{poly}(k))$. A straightforward extension to arbitrary values of Δ gives a competitive ratio of $O(\sqrt{\log k + \log\Delta})$. For the randomized case Avigdor-Elgrabli et al. [4] give an $O((\log\log(k\Delta))^2)$-competitive algorithm, i.e., an algorithm with a slight dependency on the aspect ratio of the metric space.

Gamzu and Segev [15] analyze the reordering buffer problem for n points on a line as this can be used to model the disc scheduling problem. They present a deterministic algorithm with a competitive ratio of $O(\log n)$.

In the offline case it has been shown that finding an optimal solution to the problem is NP-hard even on uniform metrics [3, 11]. Avigdor-Elgrabli and Rabani have given a constant factor approximation [5]. Im and Moseley gave an $O(\log\log(k\Delta))$-approximation for the star-metric [18] and subsequently improved this to $O(\log\log\log(k\Delta))$ [19]. Barman et al. [8] gave a bicriteria approximation algorithm that achieves an approximation guarantee of $O(\log n)$ when the buffer of the online algorithm is a constant factor *larger* than the buffer of the optimum algorithm. This works in general metric spaces.

1.2 The Model

An input sequence σ of requests has to be processed, where each request σ_i corresponds to some point in a finite metric space $M = (V, d)$. We use $n = |V|$ to denote the number of distinct points in M, and Δ to denote its *aspect ratio*, i.e., the ratio between the largest and smallest distance between two points. We assume w.l.o.g. that the minimum non-zero distance between two points is 1.

A *reordering buffer* that can store k requests can be used to rearrange the input sequence into an output sequence σ' in the following way. Initially, the buffer contains the first k requests of σ. In every time step t an algorithm has to select a request r from the buffer and append it to the output sequence, i.e., the algorithm sets σ'_t to r. If there are still requests waiting in the input sequence, the next such request takes r's place in the buffer; otherwise this place stays empty. The process is repeated until all requests from σ have been appended to the output sequence.

An online algorithm ALG has to make its decision based on the requests in the buffer and on the requests previously seen, but not based on future requests that are still to come. Suppose an algorithm ALG generates a request sequence σ' when giving a request sequence σ as input. The (true) cost $\text{ALG}_{true}(\sigma)$ is defined as

$$\text{ALG}_{true}(\sigma) = \sum_{i=1}^{\ell-1} d(\sigma'_i, \sigma'_{i+1}) \ ,$$

where ℓ is the length of the input sequence. Note that this means that the server may start its processing at the first request without incurring any cost for traveling to this location.

Throughout the paper we use a slightly different notation w.r.t. the optimum algorithm OPT for processing σ. Firstly, we assume that OPT has a larger buffer-size $h \geq k$. Secondly,

we denote the (true) cost of this algorithm with $\mathrm{OPT}(\sigma) = \mathrm{OPT}_{true}(\sigma)$. The reason is that for an online algorithm we will introduce an approximate (simplified) version of $\mathrm{ALG}_{true}(\sigma)$, which will be denoted with $\mathrm{ALG}(\sigma)$. Hence, for the online algorithm we need the differentiation between $\mathrm{ALG}(\sigma)$ and $\mathrm{ALG}_{true}(\sigma)$, whereas this is not required for the optimum algorithm.

2 The Algorithm

A *block-oriented* algorithm for the reordering buffer management problem serves requests in blocks. Whenever the buffer gets full, the algorithm identifies a set S of requests from the buffer and serves these requests. Additional requests that arrive while serving requests in S are ignored, and will not be considered until all requests in S have been handled. We call such a set S of requests chosen by the algorithm a *block*. The process of choosing and servicing blocks of requests is repeated until the end of the input sequence is reached. The requests in the buffer at this time form the last *block* of the algorithm.

For a block-oriented algorithm we can write down the (approximate) cost of the algorithm just in terms of the sequence $S_1, S_2, S_3, \ldots$ of generated blocks. This is done as follows. For a block S_i, we use $C(S_i)$ to denote the *cost of the block*, which is defined as the length of a shortest path that connects all requests in S_i (note that computing $C(S_i)$ is NP-hard). For two blocks S_i and S_j, we define the *distance* $d(S_i, S_j)$ between the two blocks, by

$$d(S_i, S_j) = \min_{r_i \in S_i, r_j \in S_j} d(r_i, r_j) \ ,$$

i.e., the distance of the closest pair $(r_i, r_j) \in S_i \times S_j$.

Suppose that for a request sequence σ a block-oriented algorithm generates a sequence $S_1, S_2, \ldots, S_\ell$ of blocks. We define the *cost* $\mathrm{ALG}(\sigma)$ of the algorithm by

$$\mathrm{ALG}(\sigma) = \sum_{i=1}^{\ell} C(S_i) + \sum_{i=1}^{\ell-1} d(S_i, S_{i+1}) \ . \tag{1}$$

We refer to the first term in Equation 1 as the *block cost* $\mathrm{ALG}_{bc}(\sigma)$ of the algorithm, and to the second term as the *connection cost* $\mathrm{ALG}_{cc}(\sigma)$. The following lemma shows that this definition of cost is close to the true cost of the algorithm. The fact that we can specify the cost of the algorithm just in terms of the generated blocks will greatly simplifies our analysis.

▶ **Lemma 1.** *We can implement any block-oriented algorithm* ALG *such that* $\mathrm{ALG}(\sigma) \leq \mathrm{ALG}_{true}(\sigma) \leq 3\,\mathrm{ALG}(\sigma)$.

Proof. In the following we describe how to serve all requests in a block S_i, and how to move to the next block S_{i+1}. Suppose the server is initially located at a request from S_i (for the first block S_1 we can assume this because according to our model the server may start at an arbitrary location). Since the requests in block S_i are known completely before serving its first request, we can efficiently compute an MST that covers all requests in S_i. The cost of traversing the elements by following the edges of the MST is at most $2C(S_i)$. Let (r_i, r_{i+1}) denote the request pair in $S_i \times S_{i+1}$ with minimum distance. We move, from our current location (after serving the last request from S_i) along a shortest path to r_{i+1}. The cost for this step is at most $C(S_i) + d(r_i, r_{i+1})$.

Repeating the above step for all blocks gives a total true cost $\mathrm{ALG}_{true}(\sigma)$ of at most $3\,\mathrm{ALG}_{bc}(\sigma) + \mathrm{ALG}_{cc}(\sigma) \leq 3\,\mathrm{ALG}(\sigma)$. ◀

In order to complete the description of the algorithm we need to describe how to choose a *good* block of requests for service when the buffer becomes full. For this we need a few definitions. For a set S of requests, and a value δ, $0 \leq \delta \leq 1$ we define the δ-fraction cost $C_\delta(S)$ of S as the minimum cost for servicing a δ-fraction of the elements from S. Formally,

$$C_\delta(S) = \min_{U \subseteq S, |U| \geq \delta \cdot |S|} C(U) \ .$$

Our algorithm StableSet is based on the following notion of a *large, stable set*. Intuitively, the cost for servicing the elements of a stable set S does not reduce by too much even if a large fraction of elements from S is removed.

▶ **Definition 2.** A set S of requests is (α, β, γ)-stable if the following holds

1. $C_{1-\alpha}(S) \geq \beta \cdot C(S)$ (stability constraint),
2. $|S| \geq \gamma k$ (size constraint).

In Section 4 we prove the following lemma showing that we can efficiently find stable sets with good parameters.

▶ **Lemma 3.** *Let V denote a set of k requests covering at most $\ell \leq k$ distinct locations in a metric space with aspect ratio Δ. There exists a polynomial time algorithm that finds an (α, β, γ)-stable subset $S \subseteq V$ with $\alpha \geq 1/(1 + \log \Delta + \log \ell)$, $\beta \geq 1/8$, and $\gamma \geq 1/e$.*

With these definitions our algorithm StableSet becomes very simple. When the buffer becomes full, choose a stable subset S from the elements of the buffer according to the algorithm implicit in Lemma 3. Then service this block of requests according to the algorithm in Lemma 1. This is repeated until the end of the input sequence is reached. The elements that still remain in the buffer form the last block of the algorithm.

3 Analysis

In the following we first describe a general approach to obtain a lower bound on the cost $\mathrm{OPT}(\sigma)$ of the optimal solution. Let $X_1, \ldots, X_\ell$ denote subsets of requests from the input sequence. We say that a subset X_i is *partially scheduled* by OPT at time t, if the first t requests in OPT's output sequence contain at least one but not all elements from X_i. The following claim gives a lower bound on the optimum cost.

▶ **Claim 4.** *Let $X_1, \ldots, X_\ell$ denote (not necessarily disjoint) subsets of requests from the input sequence, and suppose that at each point in time there are at most s subsets X_i that are partially scheduled by* OPT. *Then* $\mathrm{OPT}(\sigma) \geq \frac{1}{s} \sum_i C(X_i)$.

Proof. We associate an interval $[start(i), end(i)]$ with each set X_i, where $start(i)$ denotes the position of the first element of X_i that appears in OPT's output sequence, and $end(i)$ denotes the position of the last such element. Clearly, the cost of OPT for serving elements that lie between $start(i)$ and $end(i)$ is at least $C(X_i)$.

We can color the intervals with s colors such that intervals with the same color do not intersect. This holds because the interval graph corresponding to the set of intervals has a maximum clique size of s. Such interval graphs can be colored with s colors by a Greedy algorithm. Since sets X_i from the same color class do not interleave in OPT's output sequence the cost for serving all elements of a color-class is at least $\sum_{i \in I} C(X_i)$, where I denotes the index set of the color-class. As there must exist a color-class with cost at least $\frac{1}{s} \sum_i C(X_i)$ the claim follows. ◀

3.1 Analyzing Block Cost

The following lemma gives the bound on the block cost induced by an algorithm that uses stable sets.

▶ **Lemma 5.** *Let $S_1, S_2, \ldots, S_\ell$ denote the sequence of blocks generated by a block-oriented algorithm, where all but the last block are (α, β, γ)-stable, where $k \geq \frac{6}{\alpha\gamma}$. Then* $\mathrm{ALG}_{bc}(\sigma) \leq \left(\frac{6h}{\alpha\beta\gamma k} + 1\right) \cdot \mathrm{OPT}(\sigma)$.

Proof. Let $z = \lceil \alpha\gamma k/3 \rceil$. We obtain a set X_i $(i < \ell)$ by taking S_i and removing the first z and the last z requests from it that appear in OPT's output sequence. Then the cardinality of X_i is at least

$$|X_i| = |S_i| - 2z = |S_i| - 2\lceil \alpha\gamma k/3 \rceil \geq |S_i| - 2\alpha\gamma k/3 - 2 \geq |S_i| - \alpha\gamma k \geq (1-\alpha)|S_i| \ ,$$

where the second inequality holds for $k \geq \frac{6}{\alpha\gamma}$, and the final inequality holds due to the size constraint for block S_i. The stability constraint for S_i gives us that $C(X_i) \geq \beta C(S_i)$.

We show that at most $(h+k)/z$ sets X_i can be partially scheduled by OPT at any given time. Fix a time t. We define a partially scheduled set X_i to be of Type I if not all of S_i has already appeared in the input sequence, otherwise, we define it to be of Type II. For sets of Type II, OPT must hold the last z requests of S_i (according to the order given by OPT's output sequence) in its buffer, as these have already appeared. For sets of Type I, ALG must hold the first z requests of S_i in its buffer, as these have already appeared but ALG only starts removing elements from S_i after all of S_i has appeared. This means there can at most be k/z partially scheduled sets of Type I, and at most h/z partially scheduled sets of Type II. Applying Claim 4 gives that

$$\mathrm{OPT}(\sigma) \geq \frac{z}{h+k} \sum_{i=1}^{\ell-1} C(X_i) \geq \frac{\alpha\beta\gamma k}{6h} \sum_{i=1}^{\ell-1} C(S_i) \ . \tag{2}$$

Combining the definition of block-cost, Equation 2, and the fact that $\mathrm{OPT}(\sigma) \geq C(S_\ell)$ gives

$$\mathrm{ALG}_{bc}(\sigma) = \sum_{i=1}^{\ell} C(S_i) \geq \left(\frac{6h}{\alpha\beta\gamma k} + 1\right) \cdot \mathrm{OPT}(\sigma) \ ,$$

as desired. ◀

3.2 Analyzing Connection Cost

▶ **Lemma 6.** *Let $S_1, S_2, \ldots, S_\ell$ denote the sequence of blocks generated by a block-oriented algorithm, where all but the last block have cardinality at least γk. Then* $\mathrm{ALG}_{cc}(\sigma) \leq \left(\frac{5h}{\gamma k} + 1\right) \cdot \mathrm{OPT}(\sigma)$.

Proof. We use $\mathrm{ALG}'_{cc}(\sigma)$ to denote the connection cost, where we ignore the cost for connecting the last two blocks $S_{\ell-1}$ and S_ℓ. For every pair of consecutive blocks S_i, S_{i+1}, $i \leq \ell - 2$ we generate $\lceil \gamma k \rceil$ request-pairs by matching $\lceil \gamma k \rceil$ requests from S_i to $\lceil \gamma k \rceil$ requests from S_{i+1} in an arbitrary manner. Let X_i^r, $i \in \{1, \ldots, \ell-2\}, r \in \{1, \ldots, \lceil \gamma k \rceil\}$ denote the request pairs generated this way. We have

▶ **Fact 7.** $\sum_{i,r} C(X_i^r) \geq \gamma k \cdot \mathrm{ALG}'_{cc}(\sigma)$.

To see this, observe that for a request-pair X_i^r we have $C(X_i^r) \geq d(S_i, S_{i+1})$, as the request pair connects sets S_i and S_{i+1}. Since for every i we have at least γk requests the fact holds. The following fact allows us to apply Claim 4.

▶ **Fact 8.** *At any given time, there exist at most* $5h$ *request pairs from sets* X_i^r *that are partially scheduled by* OPT.

Proof. Fix a time step t. Suppose we have a request pair that is partially scheduled by OPT at time t. We say that it is of Type I if it has not been scheduled by ALG at all (by time step t); it is of Type II if ALG has already scheduled both requests of the pair; and it is of Type III, otherwise.

There can be at most $2h$ request pairs of Type II, because OPT must hold the second request of such a pair in its buffer, and any request can belong to at most two pairs. There can be at most $2k$ request pairs of Type I, because ALG must hold the first request of such a pair in its buffer, as this request has already appeared but ALG has not scheduled it. Finally, observe that there are at most k pairs that are partially scheduled by ALG at any point in time. Hence, the total number of partially scheduled requests of Type III is at most k.

Altogether there exists at most $3k + 2h \leq 5h$ request pairs that are partially scheduled by OPT. ◀

Combining Claim 4 with the above fact gives $\mathrm{OPT}(\sigma) \geq \frac{1}{5h} \sum_{i,r} C(X_i^r)$. Together with Fact 7 we obtain

$$\mathrm{ALG}_{cc}(\sigma) \leq \mathrm{ALG}'_{cc}(\sigma) + \mathrm{OPT}(\sigma) \leq \frac{1}{\gamma k} \sum_{i,r} C(X_i^r) + \mathrm{OPT}(\sigma) \leq \left(\frac{5h}{\gamma k} + 1\right) \cdot \mathrm{OPT}(\sigma) \ ,$$

as desired. The first inequality uses the fact that the cost for connecting the two last blocks is at most $\mathrm{OPT}(\sigma)$. ◀

3.3 Proof of the Main Result

Combining the analysis of the block cost and the connection cost gives our main theorem.

▶ **Theorem 9.** *A block-oriented algorithm that only chooses* (α, β, γ)*-stable blocks is* $O(\frac{h}{k} \cdot (\alpha\beta\gamma)^{-1})$*-competitive, against an optimal algorithm with buffer size* $h \geq k$*. Using the stable set computation from Lemma 3 gives a competitive ratio of* $O(h(\log \Delta + \min\{\log n + \log k\})/k)$ *in a metric space of* n *points and aspect ratio* Δ*.*

Proof. For the case that $k \geq \frac{6}{\alpha\gamma}$ we can simply combine the bounds in Lemma 5 and Lemma 6. For the case that $k \leq \frac{6}{\alpha\gamma}$ we use the fact that any algorithm is $O(h)$-competitive which gives the result since $O(h) = O(\frac{h}{k} \cdot k) = O(\frac{h}{k}(\alpha\beta\gamma)^{-1})$. ◀

4 Finding Stable Sets

In this section we present an algorithm for finding stable sets.

▶ **Lemma 3.** *Let* V *denote a set of* k *requests covering at most* $\ell \leq k$ *distinct locations in a metric space with aspect ratio* Δ*. There exists a polynomial time algorithm that finds an* (α, β, γ)*-stable subset* $S \subseteq V$ *with* $\alpha \geq 1/(1 + \log \Delta + \log \ell)$*,* $\beta \geq 1/8$*, and* $\gamma \geq 1/e$*.*

Proof. Set $\beta' := 1/2$ and $\alpha := 1/(1 + \log_2 \Delta + \log_2 \ell)$. For a subset S of requests we define $\mathrm{MST}_{1\text{-}\alpha}(S)$ to be a minimum spanning tree among at least $\lceil (1-\alpha)|S| \rceil$ requests from S. It is NP-hard to find such an MST but there is a 2-approximation algorithm that returns a tree $T_{1-\alpha}$ that spans $\lceil (1-\alpha)|S| \rceil$ requests and has cost $\mathrm{cost}(T_{1-\alpha}) \leq 2\,\mathrm{cost}(\mathrm{MST}_{1\text{-}\alpha}(S))$ [16].

Our algorithm for finding a stable set proceeds as follows. Initially it sets $S := V$. Then it (approximately) checks whether S is stable. For this it computes an approximation

$T_{1-\alpha}$ to $\text{MST}_{1\text{-}\alpha}(S)$ according to the algorithm by Garg [16]. Then it checks whether $\text{cost}(T_{1-\alpha}) \geq \beta' \,\text{cost}(\text{MST}(S))$. If this is the case the set S is returned. Otherwise the algorithm sets $S := V(T_{1-\alpha})$, where $V(T_{1-\alpha})$ is the vertex set of tree $T_{1-\alpha}$, and repeats the process. In the following we prove that the set S returned by the algorithm fulfills the desired constraints. We start with the stability constraint:

▶ **Fact 10.** *For each subset $U \subseteq S$, $|U| \geq (1-\alpha)|S|$, we have $C(U) \geq \frac{1}{8} \cdot C(S)$.*

Proof. We have

$$C(U) \geq \text{cost}(\text{MST}_{1\text{-}\alpha}(S)) \geq \frac{1}{2}\,\text{cost}(T_{1-\alpha}) \geq \frac{\beta'}{2}\,\text{cost}(\text{MST}(S)) \geq \frac{1}{8}C(S) \ .$$

The first step follows because an optimum path for $C(U)$ is also a spanning tree on at least $\lceil(1-\alpha)|S|\rceil$ vertices. The second step holds because of the approximation guarantee of Garg's algorithm. The third step is due to the termination condition of our procedure for finding a stable set, and the last step holds because an MST is a 2-approximation for $C(S)$. ◀

It remains to prove the size constraint. For this we require a bound on the number of iterations.

▶ **Fact 11.** *The algorithm performs at most $r_{\max} \leq \log_2(\ell\Delta) + 1$ unsuccessful iterations.*

Proof. In every unsuccessful iteration the cost of the minimum spanning tree over the set S decreases by factor $\beta' = 1/2$. The cost can be at most $\ell\Delta$ at the start, and if the cost drops below one, all remaining requests are located at a single vertex, which leads to a stable set. ◀

In every unsuccessful iteration the cardinality of the set S decreases by a $(1-\alpha)$ factor. Hence, the final cardinality is at least $k(1-\alpha)^{r_{\max}} \geq k/e$. This completes the proof of the lemma. ◀

5 Lower Bound for Memory Restricted Algorithms

In [20] Khandekar and Pandit defined a memoryless reordering buffer management algorithm as an algorithm that bases its decisions only on the content of the buffer and not on some further information that may be stored in its memory. They showed that such algorithms are severely limited by giving a lower bounds of $\Omega(k)$ on the competitive ratio. In terms of the aspect ratio their lower bound example gives $\Omega(\log\Delta/\log\log(\Delta))$.

In this section we extend their result and show that an algorithm that only bases its decision on the buffer-content and on further k bits of memory may experience a competitive ratio of $\Omega(\sqrt{\log\Delta})$. This means, if the memory used by the algorithm does not depend on the aspect ratio, the aspect ratio must appear in the competitive ratio in some form (unless, of course, the competitive ratio is a trivial bound like $O(k)$).

Since our block-oriented algorithm only needs to mark all requests that belong to the current block, it can be implemented with k bits of memory. Hence, one reason that the aspect ratio appears in our competitive ratio is the structure of the algorithm that makes it memory-restricted.

▶ **Lemma 12.** *There exists an input sequence with aspect ratio $\Delta \leq (k(k+1)2^m)^k$, for which any deterministic reordering buffer management algorithm with m bits of memory has competitive ratio $\Omega(k)$. For $m = k$ this gives a lower bound of $\Omega(\sqrt{\log\Delta})$ on the competitive ratio.*

Proof. The instance consists of a metric space over $k+1$ vertices $\{v_0, \ldots, v_k\}$, where the distance between two distinct vertices v_i and v_j is $d(v_i, v_j) = \lambda^i + \lambda^j$. The vertices can be viewed as the leaves of a star, where the vertex v_i is connected to the center of the star via an edge of length λ^i. λ will be chosen later.

Let $\mathrm{ALG}_t \in \{v_0, \ldots, v_k\}$ denote the position of the online server in the metric space after the t-th request has been served. Initially, there is a request at every vertex v_i, $i \neq 0$, and we assume contrary to the definition of our model in Section 1.2 that the online algorithm has to start at vertex v_0 (i.e., $\mathrm{ALG}_0 = v_0$). This slight change in the model does not affect our asymptotic results. The input sequence is chosen adversarily: after serving the request at ALG_t a new request at ALG_{t-1} appears. This means that whenever the online algorithm is making a decision on the next request to serve, there is a request located at every vertex v_i different from the current position of the ALG-server.

There are $k+1$ possible states of the buffer; one state for every position of the online server. In addition, the m bits of memory give rise to 2^m memory-states. The state of the algorithm is a combination of the buffer-state and the memory-state. This means in total there are $z := (k+1)2^m$ different states that the algorithm may be in. Depending on its state $\mathcal{S}$ the algorithm deterministically chooses a vertex v_{next}, and serves the request located at this vertex. Then a new request appears at its previous position, and the algorithm is in some new state $\mathcal{S}'$.

We model the servicing of our adversarial sequence σ by a deterministic algorithm, as a path on a state graph G that contains one vertex for every possible state $\mathcal{S}$, and a directed edge $(\mathcal{S}, \mathcal{S}')$ if $\mathcal{S}'$ is the successor state to state $\mathcal{S}$. We assign a weight to every edge in G as follows. If $\mathcal{S}$ corresponds to a state where the server is located at v_i and $\mathcal{S}'$ corresponds to a state with the server at position v_j, we assign a length of $d(v_i, v_j)$ to edge $(\mathcal{S}, \mathcal{S}')$. By this definition the servicing of the request sequence corresponds to a path P on the state graph and the length of this path is the cost of the online algorithm. Note that the path will actually contain a cycle C, and asymptotically the cost of the online algorithm is determined by the cost for serving the cycle.

Let $v_{i_{\max}}$ denote the vertex with largest index that corresponds to some state along the cycle, let $n_{\max}$ denote the number of states along the cycle that correspond to this position, and let n_C denote the total number of vertices along the cycle. The cost of the online algorithm for serving the cycle is at least

$$\mathrm{cost}_{\mathrm{ALG}}(C) \geq 2n_{\max}\lambda^{i_{\max}} \ ,$$

as it enters and leaves the vertex $v_{i_{\max}}$ at least $n_{\max}$ times. An optimum algorithm can serve the cycle differently. It only holds requests at location $v_{i_{\max}}$ in its buffer. All other requests are served immediately as they arrive. Then it only has to pay for the edge to $v_{i_{\max}}$ every k-th time. Hence, the optimum (average) cost for serving the cycle is at most

$$\mathrm{cost}_{\mathrm{OPT}}(C) \leq 2n_C\lambda^{i_{\max}-1} + 2n_{\max}\lambda^{i_{\max}}/k \ .$$

This gives

$$\frac{\mathrm{cost}_{\mathrm{ALG}}(C)}{\mathrm{cost}_{\mathrm{OPT}}(C)} \geq \frac{n_{\max}\lambda^{i_{\max}}}{n_C\lambda^{i_{\max}-1} + n_{\max}\lambda^{i_{\max}}/k} \geq \frac{n_{\max}}{z/\lambda + n_{\max}/k} \geq \frac{n_{\max}}{1+n_{\max}}k = \Omega(k) \ .$$

Here, we use the fact that $n_C \leq z$ (the number of states) for the second inequality, and we choose $\lambda = kz$ for the third inequality. The ratio of the costs on the cycle gives an asymptotic bound on the competitive ratio, as the cycle dominates the cost. With our choice of λ we get that the aspect ratio Δ is $\Delta \leq \lambda^k = (k(k+1)2^m)^k$. ◀

Acknowledgments. We thank Matthias Englert for making us aware that the competitive ratio of a memory restricted algorithm should depend on the aspect ratio.

References

1. Amjad Aboud. Correlation clustering with penalties and approximating the reordering buffer management problem. *Master's thesis, Computer Science Department, The Technion – Israel Institute of Technology*, 2008.
2. Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. Almost tight bounds for reordering buffer management. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 607–616, 2011.
3. Yuichi Asahiro, Kenichi Kawahara, and Eiji Miyano. NP-hardness of the sorting buffer problem on the uniform metric. *Discrete Applied Mathematics*, 160(10-11):1453–1464, 2012. doi:10.1016/j.dam.2012.02.005.
4. Noa Avigdor-Elgrabli, Sungjin Im, Benjamin Moseley, and Yuval Rabani. On the randomized competitive ratio of reordering buffer management with non-uniform costs. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 78–90, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
5. Noa Avigdor-Elgrabli and Yuval Rabani. A constant factor approximation algorithm for reordering buffer management. In *Proceedings ofproc 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 973–984, 2013. doi:10.1137/1.9781611973105.70.
6. Noa Avigdor-Elgrabli and Yuval Rabani. An optimal randomized online algorithm algorithm for reordering buffer management. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 2013.
7. Noa Avigdor-Elgrabli and Yuval Rabani. An improved competitive algorithm for reordering buffer management. *ACM Trans. Algorithms*, 11(4):1–15, June 2015. doi:10.1145/2663347.
8. Siddharth Barman, Shuchi Chawla, and Seeun Umboh. A bicriteria approximation for the reordering buffer problem. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, pages 157–168, 2012. doi:10.1007/978-3-642-33090-2_15.
9. Marcin Bienkowski, Martin Böhm, Lukasz Jez, Pawel Laskos-Grabowski, Jan Marcinkowski, Jirí Sgall, Aleksandra Spyra, and Pavel Veselý. Logarithmic price of buffer downscaling on line metrics. *CoRR*, abs/1610.04915, 2016.
10. Dan Blandford and Guy Blelloch. Index compression through document reordering. In *Proceedings of the Data Compression Conference*, DCC'02, pages 342–351, Washington, DC, USA, 2002. IEEE Computer Society.
11. Ho-Leung Chan, Nicole Megow, René Sitters, and Rob van Stee. A note on sorting buffers offline. *Theoretical Computer Science*, 423:11–18, 2012.
12. Matthias Englert and Harald Räcke. Reordering buffers with logarithmic diameter dependency for trees. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SIDA)*, pages 1224–1234, 2017.
13. Matthias Englert and Matthias Westermann. Reordering buffer management for non-uniform cost models. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 627–638, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
14. Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.
15. Iftah Gamzu and Danny Segev. Improved online algorithms for the sorting buffer problem on line metrics. *ACM Trans. Algorithms*, 6(1):15:1–15:14, December 2009.

16 Naveen Garg. Saving an ϵ: A 2-approximation for the k-MST problem in graphs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 396–402, 2005.

17 Kai Gutenschwager, Sven Spiekermann, and Stefan Voß. A sequential ordering problem in automotive paint shops. *International Journal of Production Research*, 42(9):1865–1878, 2004. doi:10.1080/00207540310001646821.

18 Sungjin Im and Benjamin Moseley. New approximations for reordering buffer management. In *Proceedings of 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1093–1111, 2014. doi:10.1137/1.9781611973402.81.

19 Sungjin Im and Benjamin Moseley. Weighted reordering buffer improved via variants of knapsack covering inequalities. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 737–748, 2015. doi:10.1007/978-3-662-47672-7_60.

20 Rohit Khandekar and Vinayaka Pandit. Online sorting buffers on line. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 584–595, 2006.

21 Jens Krokowski, Harald Räcke, Christian Sohler, and Matthias Westermann. Reducing state changes with a pipeline buffer. In *Proceedings of the 9th International Fall Workshop Vision, Modeling, and Visualization*, 2004.

22 Harald Räcke, Christian Sohler, and Matthias Westermann. Online scheduling for sorting buffers. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA)*, pages 820–832, 2002.

A Reasonable Algorithms

We define an algorithm for the reordering buffer management problem to be *reasonable* if at any point in time it chooses a request r from the buffer as the next request to be served, and then moves from its current location to r along a shortest path.

▶ **Lemma 13.** *Any reasonable algorithm for the reordering buffer management problem has competitive ratio* $2(h+k)$.

Proof. Suppose a reasonable online algorithm generates an output sequences $s_1, s_2, \ldots, s_\ell$. Its cost $\mathrm{ALG}_{true}(\sigma)$ is then $\sum_{i=1}^{\ell-1} d(s_i, s_{i+1})$. We define sets of consecutive requests: $X_i := \{s_i, s_{i+1}\}$. In the following we prove that at any point in time there can at most be $2(h+k)$ sets X_i that are partially scheduled by OPT. The result then follows by applying Claim 4.

Fix a time t. We order the elements within a request-pair X_j according to the order in which the elements are scheduled by OPT, and will refer to them as the first and second request, respectively. Suppose a request pair $X_j, j \leq t$ is partially scheduled by OPT at time t. This means that the second request of the pair must stay in OPT's buffer between steps t and $t+1$ because both requests have already appeared by time t. Note that this holds even for the case $j = t$, because the element s_{t+1} that is output by ALG at time $t+1$ must have appeared on or before time t. Any request is only contained in at most two pairs. Consequently, there can be at most $2h$ request pairs $X_j, j \leq t$, that are partially scheduled by OPT at time t.

Now, consider a request pair X_j, $j > t$ that is partially scheduled by OPT at time t. The first request of the pair is scheduled by OPT at time t or before, but ALG schedules both requests at time $t+1$ or later. Hence, the first request of the pair must be stored by ALG between steps t and $t+1$. This means we can have at most $2k$ request pairs $X_j, j > t$

that are partially scheduled by OPT at time t. In total we get at most $2(h+k)$ partially scheduled pairs. Applying Claim 4 gives

$$\mathrm{OPT}(\sigma) \geq \frac{1}{2(h+k)} \sum_i C(X_i) \ ,$$

and, hence, $\mathrm{ALG}_{true}(\sigma) \leq 2(h+k) \cdot \mathrm{OPT}(\sigma)$, as desired. ◀

Correlated Rounding of Multiple Uniform Matroids and Multi-Label Classification

Shahar Chen[1], Dotan Di Castro[2], Zohar Karnin[3], Liane Lewin-Eytan[4], Joseph (Seffi) Naor*[5], and Roy Schwartz†[6]

1 **Computer Science Department, Technion, Haifa, Israel**
shahar.chen11@gmail.com
2 **Yahoo Labs, Haifa, Israel**
dot@yahoo-inc.com
3 **Amazon, New York, NY, USA**
zkarnin@amazon.com
4 **Yahoo Labs, Haifa, Israel**
liane@yahoo-inc.com
5 **Computer Science Department, Technion, Haifa, Israel**
naor@cs.technion.ac.il
6 **Computer Science Department, Technion, Haifa, Israel**
schwartz@cs.technion.ac.il

Abstract

We introduce *correlated randomized dependent rounding* where, given multiple points $\mathbf{y}^1, \ldots, \mathbf{y}^n$ in some polytope $\mathcal{P} \subseteq [0,1]^k$, the goal is to simultaneously round each $\mathbf{y}^i$ to some integral $\mathbf{z}^i \in \mathcal{P}$ while preserving both marginal values and expected distances between the points. In addition to being a natural question in its own right, the correlated randomized dependent rounding problem is motivated by multi-label classification applications that arise in machine learning, *e.g.*, classification of web pages, semantic tagging of images, and functional genomics. The results of this work can be summarized as follows: (1) we present an algorithm for solving the correlated randomized dependent rounding problem in uniform matroids while losing only a factor of $O(\log k)$ in the distances (k is the size of the ground set); (2) we introduce a novel multi-label classification problem, the *metric multi-labeling* problem, which captures the above applications. We present a (true) $O(\log k)$-approximation for the general case of metric multi-labeling and a tight 2-approximation for the special case where there is no limit on the number of labels that can be assigned to an object.

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

Keywords and phrases approximation algorithms, randomized rounding, dependent rounding, metric labeling, classification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.34

1 Introduction

Randomized rounding [32] is a fundamental technique in approximation algorithms. In this approach, given a solution $\mathbf{y} \in \mathbb{R}^k$ to some linear program, each y_i is *independently* rounded

* Joseph (Seffi) Naor's work is supported in part by ISF grant 1585/15 and US-Israel BSF grant 2014414 (part of this work was done while visiting the Simons Institute for the Theory of Computing).
† Roy Schwart'z work is supported by ISF grant 1336/16.

into an integral value. Unfortunately, when constraints on the rounded solution are present, randomized rounding does not always produce a feasible solution. Hence, *dependent* rounding schemes were introduced [1, 2, 10, 12, 22, 26, 35]. In general, dependent rounding needs to solve the following problem: given a polytope $\mathcal{P} \subseteq [0,1]^k$ over ground set K of size k and $\mathbf{y} \in \mathcal{P}$, round $\mathbf{y}$ into $\mathbf{z} \in \mathcal{P} \cap \{0,1\}^k$ such that $\mathbb{E}[\mathbf{z}] = \mathbf{y}$. Intuitively, $\mathbf{z}$ has the following two properties: (1) $\mathbf{z}$ is always integral and feasible since $\mathbf{z} \in \mathcal{P} \cap \{0,1\}^k$; and (2) $\mathbf{z}$ preserves the marginal values given by $\mathbf{y}$ for each element in K since $\mathbb{E}[\mathbf{z}] = \mathbf{y}$. The above problem has been extensively studied and was solved for different types of polytopes $\mathcal{P}$, *e.g.*, bipartite matching and b-matching [22, 26], uniform matroids [35], spanning trees [2], and general matroids [12][1].

In this work we consider a natural extension of dependent rounding in which we are given many points in $\mathcal{P}$ and the goal is to round all the points, while preserving both marginal values and expected distances (up to some loss) between any pair of points. Formally, given a polytope $\mathcal{P} \subseteq [0,1]^k$ over ground set K of size k and $\mathbf{y}^1, \ldots, \mathbf{y}^n \in \mathcal{P}$, we need to round each $\mathbf{y}^i$ to some $\mathbf{z}^i$ such that the following hold: (1) $\mathbf{z}^i \in \mathcal{P} \cap \{0,1\}^k$ for every $i = 1, \ldots, n$; (2) $\mathbb{E}\left[\mathbf{z}^i\right] = \mathbf{y}^i$ for every $i = 1, \ldots, n$; and (3) there exists some loss factor α such that $\mathbb{E}\left[||\mathbf{z}^i - \mathbf{z}^j||_1\right] \leq \alpha ||\mathbf{y}^i - \mathbf{y}^j||_1$ for every $i, j = 1, \ldots, n$. We call this problem *correlated randomized dependent rounding*. Note that requirements (1) and (2) imply that each $\mathbf{z}^i$ is a feasible rounding of $\mathbf{y}^i$ that preserves marginal values, as in the standard dependent rounding setting. The novelty of our problem lies in requirement (3) which states that for all pairs of points the expected distance after the rounding, *i.e.*, $\mathbb{E}\left[||\mathbf{z}^i - \mathbf{z}^j||_1\right]$, is within a factor of α from the original distance between the points, *i.e.*, $||\mathbf{y}^i - \mathbf{y}^j||_1$. Additionally, it will be useful also to consider an extension of the above where each point $\mathbf{y}^i$ (and thus also $\mathbf{z}^i$) is required to be in a different polytope $\mathcal{P}_i$.

Our main reason for introducing the correlated randomized dependent rounding setting originates from multi-label classification problems. In classification problems, one must assign labels to objects given some observed data. In this work we consider classification problems where multiple labels can be assigned to each object. Such problems naturally arise in various settings, *e.g.*, classification of textual data such as web pages [38, 39], semantic tagging of images and videos [7, 30, 42], and functional genomics [4, 5].

The assignment of labels to objects should be done in a manner that is most consistent with the observed data, from which two important ingredients are derived. The first is an *assignment cost* for every (object,label) pair, reflecting a recommendation given by a local learning process which infers label preferences of objects. The second is similarity information on pairs of objects, giving rise to *separation costs* incurred once different label sets are assigned to a pair of similar objects. Our goal is to find a labeling that minimizes a global cost function, while taking into account both local and pairwise information.

To provide some intuition for the formal problem given below and the possible range of its parameters, we provide a concrete example. The objective in the example is that of assigning topics to web pages, where objects are the web pages and labels are the topics. Here, it is very natural for a web page to discuss more than one topic. The assignment cost of a (webpage,topic) pair can be derived from the features associated with a web page, *e.g.*, its words, or shingles, and the domain it is located in. However, consider information from search queries leading to the web page. A specific search query is typically observed only a

[1] In some of the above works, additional properties of $\mathbf{z}$ are required, *e.g.*, concentration of linear functions over $\mathbf{z}$. Since such concentration bounds are not required for the metric multi-labeling (MML) problem, the discussion on this topic is postponed to a full version of the paper.

handful of times, and though features can be extracted from it, a very natural way to use the latter information is by having pairwise similarity relations between web pages, if both were reached by the same search query.

We note that when assigning multiple labels to objects, it is often desirable to bound the number of labels assigned to objects. As a matter of fact, in most of the papers cited above the total number of labels can be in the thousands or even millions, while each object is expected to be assigned only a handful of labels. In particular, in the above example, we expect a single webpage to be assigned only a small fraction of all possible topics. This property imposes further constraints on our objective that we elaborate on below.

We are now ready to introduce the *metric multi-labeling* (MML) problem. In (MML) we are given a set of nodes V, where each node corresponds to an object, and a set of labels $K = \{1, 2, \ldots, k\}$. The pairwise relations are given in the form of an edge set E and a weight function $s : E \to \mathbb{R}^+$, capturing similarity between objects. Additionally, the bound function $b : V \to \mathbb{N}$ specifies how many labels can be assigned to each node. Finally, we are given an assignment cost function $c : V \times K \to \mathbb{R}$. Assignment costs may be either positive or negative, reflecting a recommendation given by a local learning process which infers the label preferences of objects. Intuitively, if $c(v, \ell) \geq 0$ (or $c(v, \ell) < 0$) we say that node v *dislikes* (or *likes*) label ℓ. A detailed explanation as to why assignment costs might be either positive or negative is deferred to a full version of the paper. The learning process determining assignment costs ignores pairwise relations between objects. Clearly, the labeling cost of completely agreeing with this recommendation is the minimum possible, and this is our *benchmark labeling*. We evaluate the assignment cost of a labeling by its deviation from the benchmark labeling.

A feasible multi-labeling $f : V \to 2^K \setminus \emptyset$ is an assignment of at least one label to every node, such that $|f(v)| \leq b_v$, *i.e.*, the number of labels assigned to v is at most b_v. For the special case where $b_v = k$ for every $v \in V$, *i.e.*, there is no upper bound on the number of labels that can be assigned to a node, we denote the problem by (Unbounded-MML).

The cost of a multi-labeling is measured by the sum of two terms: assignment costs and separation costs. Let us first focus on assignment costs, which measure the deviation of f from the benchmark labeling. Specifically, for every node v, the benchmark labeling assigns to v all labels it likes, *i.e.*, labels ℓ for which $c(v, \ell) < 0$, and does not assign to v any of the labels it dislikes, *i.e.*, labels ℓ for which $c(v, \ell) \geq 0$. Thus, focusing on a single label ℓ, f deviates from the benchmark labeling by $c(v, \ell)$ if $\ell \in f(v)$ and ℓ is a label v dislikes, *i.e.*, $c(v, \ell) \geq 0$. Similarly, f deviates from the benchmark labeling by $|c(v, \ell)|$ if $\ell \notin f(v)$ and ℓ is a label v likes, *i.e.*, $c(v, \ell) < 0$. Formally, denote by $K^+(v) \triangleq \{\ell \in K : c(v, \ell) \geq 0\}$ the collection of all labels v dislikes, and by $K^-(v) \triangleq \{\ell \in K : c(v, \ell) < 0\}$ the collection of all labels v likes. Then, the total assignment cost of node v with respect to f is: $\sum_{\ell \in K^+(v)} c(v, \ell) \mathbf{1}_{\{\ell \in f(v)\}} + \sum_{\ell \in K^-(v)} |c(v, \ell)| \, \mathbf{1}_{\{\ell \notin f(v)\}}$.

Let us now focus on the separation costs. The separation cost of edge (u, v) is the number of labels nodes u and v disagree on, *i.e.*, the ℓ_1 distance between the characteristic vectors of $f(u)$ and $f(v)$. Formally, a pair of nodes (u, v), given a multi-labeling f, incurs the following separation cost: $s(u, v) \cdot ||\mathbf{1}_{f(u)} - \mathbf{1}_{f(v)}||_1$. For any subset of labels $S \subseteq K$, $\mathbf{1}_S$ denotes the characteristic vector of S. Summing up over the above we are now ready to provide a formal

definition of the (MML) problem: find a feasible multi-labeling f that minimizes

$$\sum_{v\in V}\left(\sum_{\ell\in K^+(v)} c(v,\ell)\mathbf{1}_{\{\ell\in f(v)\}} + \sum_{\ell\in K^-(v)} |c(v,\ell)|\,\mathbf{1}_{\{\ell\notin f(v)\}}\right) + \sum_{(u,v)\in E} s(u,v)||\mathbf{1}_{f(u)} - \mathbf{1}_{f(v)}||_1\,. \tag{1}$$

Summarizing, (MML) is a novel classification model in which multiple labels can be assigned to objects. We emphasize that in (MML), obtaining a solution to the (global) optimization objective is decoupled from the local learning process for the objects, thus allowing us to view the output of these processes as part of the input to (MML), and treating them in a "black box" fashion.

Let us now focus on our results. We introduce the correlated randomized dependent rounding problem and the (MML) problem. We tackle the correlated dependent rounding problem for the case of multiple (possibly different) uniform matroids, as summarized in the following theorem.

▶ **Theorem 1.** *Let K be a ground set of size k and $M_1,\ldots,M_n$ be n uniform matroids over K, where $rank(M_i) = b_i$. Additionally, let $\mathbf{y}^i \in \{\mathbf{y}\in[0,1]^k : \sum_{\ell=1}^k y_\ell \le b_i\}$ for every $i=1,\ldots,n$. Then there is an efficient algorithm for sampling $\mathbf{z}^1,\ldots,\mathbf{z}^n$ s.t.: (1) $\mathbf{z}^i$ is the characteristic vector of an independent set of M_i for every $i=1,\ldots,n$; (2) $\mathbb{E}\left[\mathbf{z}^i\right] = \mathbf{y}^i$ for every $i=1,\ldots,n$; and (3) $\mathbb{E}\left[||\mathbf{z}^i - \mathbf{z}^j||_1\right] \le O(\log k)||\mathbf{y}^i - \mathbf{y}^j||_1$ for every $i,j=1,\ldots,n$.*

Note that the loss in the distance, *i.e.*, property (3) above, depends only on the size of the ground set k and not on the number of given matroids n.

We use the above to obtain a (true) approximation of $O(\log k)$ for (MML). For the special case of (Unbounded-MML) we present a tight 2-approximation.

▶ **Theorem 2.** *The (MML) problem admits a (true) approximation of $O(\log k)$.*

▶ **Theorem 3.** *The (Unbounded-MML) problem admits an approximation of 2.*

▶ **Theorem 4.** *Assuming the unique games conjecture, the (Unbounded-MML) problem does not admit an approximation better than $2\,(1-1/k)$.*

Let us now focus on our approach and techniques. Consider the correlated dependent rounding problem, we now elaborate as to why known techniques fail when applied to it. The problem of rounding of online paging [6] is closely related to correlated dependent rounding. Unfortunately, techniques developed in the paging context allow us to bound distances only between *some* of the pairs of points, *i.e.*, $\mathbb{E}\left[||\mathbf{z}^{i+1} - \mathbf{z}^i||_1\right]$ for every $i=1,\ldots,n-1$, as opposed to the desired $\mathbb{E}\left[||\mathbf{z}^i - \mathbf{z}^j||_1\right]$ for every $i,j=1,\ldots,n$. Therefore, a different approach is required.

We note that achieving requirements (1) and (2) alone, *i.e.*, $\mathbf{z}^i \in \mathcal{P}\cap\{0,1\}^k$ and $\mathbb{E}\left[\mathbf{z}^i\right] = \mathbf{y}^i$ for every $i=1,\ldots,n$, has already been achieved by any of the dependent rounding algorithms that can be applied to a uniform matroid, *e.g.*, [10, 12, 35] (just execute the algorithm independently for each $\mathbf{y}^i$). Obviously, this approach completely fails when considering requirement (3), *i.e.*, $\mathbb{E}\left[||\mathbf{z}^i - \mathbf{z}^j||_1\right] \le \alpha||\mathbf{y}^i - \mathbf{y}^j||_1$, as α might be unbounded. The reason for the latter is that if $\mathbf{y}^i = \mathbf{y}^j$ for some $i \ne j$, then $||\mathbf{y}^i - \mathbf{y}^j||_1 = 0$ but $\mathbb{E}\left[||\mathbf{z}^i - \mathbf{z}^j||_1\right] > 0$ (as the two executions of dependent rounding, one for $\mathbf{y}^i$ and the other for $\mathbf{y}^j$, are independent).

Our approach to solving the above is to correlate all n executions of dependent rounding, one for each $\mathbf{y}^1, \ldots, \mathbf{y}^n$. Specifically, we execute the randomized dependent rounding algorithm of [35] for each $\mathbf{y}^i$ separately, but use the *same* random bits as input for all n different executions. Remarkably, this simple approach suffices. However, we note that the analysis of our algorithm uses the specific inner-workings of the algorithm of [35]. Hence, it seems that correlated dependent rounding cannot be easily solved through a "black box" application of any dependent rounding algorithm, *e.g.*, [10, 12].

Let us now focus on the special case (Unbounded-MML) and illustrate why known algorithms and techniques fail when applied to it. (Unbounded-MML) is inspired by the *metric labeling* problem, first introduced in full generality by [25]. In the metric labeling problem we are given an edge weighted graph $G = (V, E)$, a collection K of k labels, a *non-negative* assignment cost function $c : V \times K \to \mathbb{R}^+$, and a metric d over K. The goal is to assign a *single* label to each node while minimizing the sum of assignment and separation costs. As in (Unbounded-MML), assignment costs are defined using c, whereas the separation cost of edge (u, v) is the distance in the metric d between the labels assigned to u and v. It is important to note that metric labeling differs from (Unbounded-MML) in two main points: (1) each object can be assigned exactly one label, as opposed to multiple labels in (Unbounded-MML), and (2) the assignment cost function c is non-negative, whereas in (Unbounded-MML) assignment costs may be either positive or negative.

Consider a further restricted special case of (Unbounded-MML) where all assignment costs are non-negative. If one applies the algorithm of [25] by expanding the label set K to $2^K \setminus \emptyset$ and considering the ℓ_1 metric on the expanded set[2], then this not only results in a large approximation guarantee of $O(k)$, but also the running time of the algorithm scales with 2^k and not k. More generally, we wish to claim that existing techniques and algorithms for the metric labeling problem cannot be directly applied to (Unbounded-MML). Consider a node v which has multiple labels ℓ it likes, *i.e.*, $c(v, \ell) < 0$. Since only a single label is allowed per node in metric labeling, it must be the case that whatever algorithm or technique we use, there is at least one label v likes that ultimately is not assigned to v. Thus, potentially incurring a huge loss in the objective.

We address the above difficulties by employing two approaches. First, we use a *global* charging argument over all labels in K when bounding the separation cost of an edge (u, v). Typically, such global arguments are avoided, *e.g.*, all known algorithm for metric labeling (either with a general or a specific metric) do not employ any type of global argument. Second, we *distort* the optimal marginal probabilities $x_{v,\ell}$ given by the linear programming relaxation for (Unbounded-MML). This enables us to balance both positive and negative assignment costs, along with separation costs.

Let us now mention some related work. An extensively studied topic is that of dependent rounding of fractional solution. A randomized variant of pipage rounding [1] was given by [22] who applied it to assignment polytopes (see also [26, 35]). An approach based on maximum entropy for dependent rounding was introduced by [3] in the context of max-min allocations, and was later extended to spanning trees by [2]. When considering general matroid independence polytopes, [10, 12] provided methods of conducting dependent rounding.

(MML) gets as input costs for assigning labels to objects and a similarity measure between objects. The labeling costs are based on a multi-label learning process (supervised learning) which is applied to a set of instances, each belonging potentially to multiple classes (labels),

[2] Only the general algorithm of [25] is known for the case of ℓ_1 distances over the k-dimensional hypercube, and it achieves an approximation of $O(k)$. This guarantee is tight as it based on tree metrics.

and predicts a set of class labels given a new instance. *Multi-label classification* has attracted much attention following various real world problems requiring usage of multiple labels [37], and thorough surveys in this area can be found in [34, 36]. The basic approach transforms the original problem into several instances of simpler binary classification problems, where each instance corresponds to a single label. This method is called *binary relevance*, and it assumes that labels are independent of each other, and thus one needs to solve k separate binary-label classification problems, where k denotes the number of labels. Approaches based on classifier chains have been adopted to model interdependencies between labels while maintaining acceptable computational complexity [33].

The *label power set* approach transforms the problem into a multi-class problem [14], where labels in the multi-class problem are a cross product of the original labels (and cover all possible combinations of these labels), resulting in the problem of mapping each data point to a binary vector. The main drawback of this approach is poor scaling in terms of the number of labels (e.g., vision problems where the number of categories may be large). A different approach addresses the problem directly, in its full generality, and is much harder than the traditional binary and multi-class problems, which in fact are special cases of multi-labeling. Some notable examples of multi-label algorithms, which are extensions based on binary problems, are adaptations of AdaBoost [21], the *ML-kNN* [41] based on kNN algorithm [20], and *Clare* which is an adapted decision tree algorithm for multi-label classification [31].

Another related machine learning approach is *kernel pairwise classification* [40]. Here, relations between pairs of samples are given using kernels. Supervised pairwise prediction aims to predict such pairwise relationships based on known relationships. Pairwise prediction takes a pair of instances as its input, and outputs the relationship between the two instances. The application of kernel methods to pairwise classification is based on a kernel function between two pairs of instances [24]. The main difference between this approach and our setting is that it does not consider single items, but rather focuses only on pairwise relations.

Metric labeling is an elegant and powerful mathematical model capturing a wide range of classification problems, where information about objects, as well as their pairwise relations, is given. Notice that such a scenario is not captured by neither known multi-class classification techniques, nor by existing pairwise kernel based techniques. The problem was first formulated in full generality by [25], and captures many classification problems that arise in various settings. Specifically, metric labeling has applications in important fields such as Markov theory [13, 27], image processing and computer vision [18, 8], as well as language modeling [29]. In [25], the authors gave an $O(\log k)$-approximation for any metric[3], and a 2-approximation for the uniform metric case. The latter is known to be tight assuming the unique games conjecture [28]. It is worth mentioning that metric labeling is of much importance in the combinatorial optimization setting, as it captures well studied problems such as multiway cut [9, 15, 16, 17, 23] and 0-extension [11, 19].

2 Preliminaries

We formulate the following natural linear programming relaxation for the (MML) problem (similarly to the relaxation given by [25] for uniform metric labeling). Variable $x_{v,\ell}$ is the (fractional) indicator for labeling node v with label ℓ. The first constraint guarantees that each node v receives between 1 and b_v labels. The following two constraints, along with the fact that the problem is a minimization problem, imply that $z_{u,v,\ell} = |x_{u,\ell} - x_{v,\ell}|$, i.e.,

[3] The metric over the labels determines their pairwise distances and can be arbitrary in general.

$z_{u,v,\ell}$ is the separation cost of nodes u and v with respect to label ℓ. Hence, the fourth constraint asserts that $d_{u,v}$ equals $||\mathbf{x}_u - \mathbf{x}_v||_1$, where $\mathbf{x}_u = (x_{u,1}, \ldots, x_{u,\ell})$ for every $u \in V$. The objective of the relaxation follows directly from the definition of (MML) (1).

$$
\begin{aligned}
\min \quad & \textstyle\sum_{v\in V}\left[\sum_{\ell\in K^+(v)} c(v,\ell)x_{v,\ell}+\sum_{\ell\in K^-(v)}|c(v,\ell)|(1-x_{v,\ell})\right]+\sum_{u,v\in V} s(u,v)d_{u,v} \\
s.t. \quad & 1 \le \sum_{\ell\in K} x_{v,\ell} \le b_v && \forall v \in V \\
& z_{u,v,\ell} \ge x_{u,\ell} - x_{v,\ell} && \forall u,v \in V \\
& z_{u,v,\ell} \ge x_{v,\ell} - x_{u,\ell} && \forall u,v \in V \\
& d_{u,v} = \sum_{\ell\in K} z_{u,v,\ell} && \forall u,v \in V \\
& 0 \le x_{v,\ell} \le 1 && \forall v \in V, \forall \ell \in K
\end{aligned}
$$

The following observation simplifies the analysis of the separation cost considerably.

▶ **Observation 5.** *Without loss of generality we can simply assume that any two adjacent nodes differ in only a single coordinate, by a value $\varepsilon > 0$, which can be made arbitrarily small. Specifically, given $(u,v) \in E$ we assume that $\mathbf{x}_u = (x_{u,1}, x_{u,2}, \ldots, x_{u,k})$ and $\mathbf{x}_v = (x_{u,1} + \varepsilon, x_{u,2}, \ldots, x_{u,k})$.*

3 Correlated Randomized Dependent Rounding

Denote by M_{K,b_v} the uniform matroid over K of rank b_v, and recall that $\mathcal{P}(M_{K,b_v}) = \{\mathbf{x} \in [0,1]^k : \sum_{\ell=1}^{k} x_\ell \le b_v\}$ is the standard independent set polytope corresponding to M_{K,b_v}. For completeness, we start by presenting the basic building block of [35] for rounding a single point in $\mathcal{P}_{M_{K,b_v}}$, as we later require its inner-workings.

Let us now focus on rounding a single point in the uniform matroid polytope. The basic building block (Algorithm 1) receives two marginal probabilities, $0 \le \alpha \le 1$ and $0 \le \beta \le 1$, for the i^{th} and j^{th} labels correspondingly, and randomly updates them. At least one of the updated marginal probabilities, denoted by α' and β', is "rounded" to either 0 or 1. This is done while deterministically preserving the sum of the marginal probabilities, and each of the marginal probabilities is preserved in expectation. Lemma 6 summarizes the above, and its proof is deferred to a full version of the paper.
It is important to note that $0 \le \alpha', \beta' \le 1$ always, *i.e.*, Algorithm 1 returns valid marginal probabilities.

▶ **Lemma 6.** *Upon the termination of Algorithm 1:*
1. $\mathbb{E}[\alpha'] = \alpha$ *and* $\mathbb{E}[\beta'] = \beta$.
2. $\alpha' + \beta' = \alpha + \beta$ *always.*
3. *One of i and j is declared* fixed *and its marginal value belongs to* $\{0,1\}$.

Define a *label tree* T of K to be a full binary tree with exactly k leaves, where each leaf corresponds to a distinct label of K. We now describe the rounding procedure which we denote by *label tree rounding*. It receives as input a label tree T, a point $\mathbf{x}_v \in \mathcal{P}(M_{K,b_v})$, a collection of independent random thresholds $\theta_z \sim Unif[0,1]$ for every non-leaf node z of T, and one additional independent random threshold $\theta \sim Unif[0,1]$. The label tree rounding procedure operates as follows:
1. Every leaf of T sends to its parent its label and its marginal value as given by the relaxation, *i.e.*, a leaf that corresponds to label $\ell \in K$ sends to its parent $(\ell, x_{v,\ell})$.

Algorithm 1 Resolve(i, α, j, β).

draw a threshold $\theta \sim Unif[0,1]$.
if (case (a)) $0 \leq \alpha + \beta \leq 1$ **then**
  if $\theta \leq {}^{\alpha}/_{(\alpha+\beta)}$ **then**
    $\alpha' \leftarrow \alpha + \beta$, $\beta' \leftarrow 0$, and $s \leftarrow j$.
  else
    $\alpha' \leftarrow 0$, $\beta' \leftarrow \alpha + \beta$, and $s \leftarrow i$.
  end if
end if
if (case (b)) $1 < \alpha + \beta \leq 2$ **then**
  if $\theta \leq {}^{(1-\beta)}/_{(2-\alpha-\beta)}$ **then**
    $\alpha' \leftarrow 1$, $\beta' \leftarrow \alpha + \beta - 1$, and $s \leftarrow i$.
  else
    $\alpha' \leftarrow \alpha + \beta - 1$, $\beta' \leftarrow 1$, and $s \leftarrow j$.
  end if
end if
return (i, α', j, β') and declare s as *fixed*.

2. Every non-leaf node z of T (that is not the root) receives from its two children (i, α) and (j, β); it executes Algorithm 1 with parameters (i, α, j, β) and θ_z as the random threshold to obtain (α', β'); updates the marginal probabilities of i and j to be α' and β' respectively; and sends to its parent in T the label that was *not* fixed from $\{i, j\}$ along with its newly updated marginal probability.
3. The root r of T operates exactly as any other non-leaf node of T with the following exception: instead of sending the label that was not fixed to its parent along with its newly updated marginal probability, r uses the given random threshold θ to round the label that was not fixed, *i.e.*, after the execution of Algorithm 1 by r if $s \in \{i, j\}$ denotes the label that is not fixed and the newly updated marginal probability of s equals γ, then r sets the marginal of s to be 1 if $\theta \leq \gamma$ and 0 otherwise.

The following lemma summarizes the desired properties of the label tree rounding procedure, and its proof is deferred to a full version of the paper.

▶ **Lemma 7.** *Let $v \in V$, $\mathbf{x}_v \in \mathcal{P}(M_{K,b_v})$, T a label tree of K, and denote by $\tilde{\mathbf{x}}_v$ the vector of marginal probabilities obtained by executing the label tree rounding procedure. Then,*
1. $\tilde{\mathbf{x}}_v \in \{0,1\}^k$.
2. *Let $B_v \triangleq \sum_{\ell \in K} x_{v,\ell}$, then $\lfloor B_v \rfloor \leq \sum_{\ell \in K} \tilde{x}_{v,\ell} \leq \lceil B_v \rceil$ always.*
3. *For every $\ell \in K$: $\Pr[\tilde{x}_{v,\ell} = 1] = x_{v,\ell}$.*

Let us now focus on rounding multiple points in the uniform matroid polytope. In this section we describe how to round multiple points in $\mathcal{P}(M_{K,b_v})$ while: (1) preserving marginal probabilities; and (2) being "faithful" to the original ℓ_1 distances between any pair of points in $\mathcal{P}(M_{K,b_v})$. Our correlated rounding procedure receives as input a fixed label tree T, along with n points $\{\mathbf{x}_v\}_{v \in V}$ in $\mathcal{P}(M_{K,b_v})$. Intuitively, it applies the label tree rounding procedure to all n points simultaneously, while using the same given tree T and the same random thresholds in all executions. A formal description appears in Algorithm 2. As before, we denote by $\tilde{\mathbf{x}}_v$ the output of Algorithm 2 for node $v \in V$.

Lemma 8 bounds the expected separation cost of neighbouring nodes u and v. Assuming $\mathbf{x}_u$ and $\mathbf{x}_v$ differ only in label 1 (as Observation 5 states without loss of generality), the

Algorithm 2 Correlated Rounding$\left(\{\mathbf{x}_v\}_{v\in V}, T\right)$.

For every non-leaf node z of T draw an independent threshold $\theta_z \sim Unif[0,1]$.
Draw an independent threshold $\theta \sim Unif[0,1]$ for the root r of T.
$\forall v \in V$: execute label tree rounding with input T, $\mathbf{x}_v$, $\{\theta_z\}_{z \text{ non-leaf node of } T}$, and θ.
Output the resulting $\{\tilde{\mathbf{x}}_v\}_{v\in V}$.

executions of Algorithm 1 can differ between u and v only at non-leaf nodes of T that lie on the (single) path from the leaf that represents label 1 and the root r of T. At the heart of the proof lies the following observation: the expected *additive* increase in $||\mathbf{x}_u - \mathbf{x}_v||_2$ is $O(\varepsilon)$ for each of the non-leaf nodes of T that lie on the above mentioned path.

▶ **Lemma 8.** *Let $u, v \in V$ be such that $\mathbf{x}_u$ and $\mathbf{x}_v$ satisfy Observation 5, let $\tilde{\mathbf{x}}_u$ and $\tilde{\mathbf{x}}_v$ be the output of Algorithm 2 for nodes u and v correspondingly, and let δ be the depth of T. Then,*

$$\mathbb{E}\left[||\tilde{\mathbf{x}}_u - \tilde{\mathbf{x}}_v||_1\right] \leq O(\delta)\varepsilon\,.$$

Proof. Recall that Observation 5 states that $\mathbf{x}_u$ and $\mathbf{x}_v$ are identical, except that $x_{v,1} = x_{u,1} + \varepsilon$. Hence, let P be the path from the leaf in T representing label 1 to the root r of T, and denote the sequence of nodes in this path by $z_1, z_2, z_3 \ldots, z_m$ (where z_1 is the leaf and z_m is the root r). We use the following two assumptions that can be made without loss of generality.

First, as the order of executions of Algorithm 1 at the nodes of T is irrelevant to the outcome of Algorithm 2, as long as execution of Algorithm 1 at some node z of T is performed after all executions of Algorithm 1 at all non-leaf nodes in the induced subtree of T that z is its root. Hence, let us assume without loss of generality that all executions of Algorithm 1 at nodes not in P are performed before any execution of Algorithm 1 at nodes $z_2, z_3, \ldots, z_m$.

Second, note that in every non-leaf node along P, *i.e.*, $z_2, z_3, \ldots, z_m$, exactly one execution of Algorithm 1 is performed for each of the nodes u and v. The execution of Algorithm 1 at some node z_p, $p = 2, \ldots, m$, receives exactly two labels as input, one from the child z_{p-1} (along the path P) and the other from the other child of z_p which we denote by w_{p-1} (not on the path P). It is important to note that each of these two inputs might be random, however, the input received from node w_{p-1} is *always identical* for both u and v. Therefore, let us denote for simplicity of presentation and without loss of generality that the input w_{p-1} sends to the execution of Algorithm 1 at node z_p is label number p with its updated marginal probability γ_p, *i.e.*, (p, γ_p). Thus, we can focus only on the first m labels of K since for labels $m+1, \ldots, k$ nodes u and v will always be identical and their contribution to $||\tilde{\mathbf{x}}_u - \tilde{\mathbf{x}}_v||_1$ will be always 0.

Denote by $\mathbf{x}_u^t \in [0,1]^m$ and $\mathbf{x}_v^t \in [0,1]^m$ the vector of marginal probabilities of the first m labels after performing the execution of Algorithm 1 at node z_t, for nodes u and v respectively. Thus, for example, $\mathbf{x}_u^1 = (u_1, \gamma_2, \gamma_3, \ldots, \gamma_m)$ and $\mathbf{x}_v^1 = (u_1 + \varepsilon, \gamma_2, \gamma_3, \ldots, \gamma_m)$, and $\mathbf{x}_u^m = (\tilde{x}_{u,1}, \tilde{x}_{u,2}, \tilde{x}_{u,3}, \ldots, \tilde{x}_{u,m})$ and $\mathbf{x}_v^m = (\tilde{x}_{v,1}, \tilde{x}_{v,2}, \tilde{x}_{v,3}, \ldots, \tilde{x}_{v,m})$. We prove that:

$$\mathbb{E}\left[||\mathbf{x}_u^t - \mathbf{x}_v^t||_1 - ||\mathbf{x}_u^{t-1} - \mathbf{x}_v^{t-1}||_1\right] \leq 2\varepsilon \qquad \forall t = 2, 3, \ldots, m. \tag{2}$$

The proof of the lemma is completed by summing (2) over all relevant values of t, and recalling that $||\mathbf{x}_u^1 - \mathbf{x}_v^1||_1 = ||\mathbf{x}_u - \mathbf{x}_v||_1 = \varepsilon$. Inequality (2) is proved by examining the joint distribution of Algorithm 1 at node z_t for both u and v. This computation is deferred to a full version of the paper. ◀

Algorithm 3 Single Threshold (ST).

draw a threshold $\theta \sim Unif[0,1]$.
for every $v \in V$ **do**
  $f_{ST}(v) \leftarrow \{\ell : \theta \leq h(x_{v,\ell})\}$.
end for
output f_{ST}.

Algorithm 4 Kleinberg-Tardos (KT).

while $V \neq \emptyset$ **do**
  independently draw a threshold $\theta \sim Unif[0,1]$ and a uniform label $\ell \in K$.
  for every $v \in V$ **do**
    if $\theta \leq x_{v,\ell}$ **then**
      $f_{KT}(v) \leftarrow \{\ell\}$ and $V \leftarrow V \setminus \{v\}$.
    end if
  end for
end while
output f_{KT}.

Proof (of Theorem 1). Apply Algorithm 2 to the given points $\mathbf{y}^1, \ldots, \mathbf{y}^n$ with a label tree T whose depth is $O(\log k)$. Lemmas 7 and 8 conclude the proof. ◀

4 $O(\log k)$-Approximation for MML

Proof (of Theorem 2). We apply Algorithm 2 to the fractional solution provided by the linear programming relaxation. Starting with assignment costs, Property (3) of Lemma 7 implies that all assignment costs are preserved in expectation. Considering separation costs, one can always choose a label tree T whose depth is $O(\log k)$, and thus Lemma 8 implies a multiplicative loss of $O(\log k)$ in the separation costs. Finally, Property (2) of Lemma 7 guarantees that every node $v \in V$ is assigned at most $\lceil B_v \rceil$ labels, but since $\lceil B_v \rceil \leq b_v$ our algorithm never deviates from the bound on the number of labels. Additionally, it is important to note that Property (2) of Lemma 7 also ensures that every node $v \in V$ is assigned at least one label since $\lfloor B_v \rfloor \geq 1$. ◀

5 A Tight Approximation for Unbounded MML

Let us now focus on the basic building blocks. We use the following two algorithms as basic building blocks for our final algorithm. The first is a simple single threshold algorithm. Let $h : [0,1] \to [0,1]$ be a monotone non-decreasing *distortion* function. The algorithm applies the distortion function h to each fractional value $x_{v,\ell}$, and then finds a multi-labeling $f_{ST}(v) : V \to 2^K$ by assigning to v all labels ℓ whose *distorted* fractional value is larger than a uniformly random threshold $\theta \in [0,1]$. The choice of an appropriate distortion function h plays a crucial role in obtaining the best possible approximation of 2 for (Unbounded-MML).

The second building block we use is due to [25]. It is the 2-approximation they provide for the uniform metric labeling problem.

Our algorithm is a simple "merge" of the two basic building blocks: Algorithms 3 and 4 are run independently and the union of their label assignments is returned.

Algorithm 5 Union.

independently run Algorithms 3 and 4 to obtain f_{ST} and f_{KT}.
for every $v \in V$ **do**
  $f(v) \leftarrow f_{ST}(v) \cup f_{KT}(v)$.
end for
output f.

Let us focus on the assignment costs. We denote by $\mathbf{x}_v \in [0,1]^k$ the vector corresponding to v, i.e., $(\mathbf{x}_v)_\ell = x_{v,\ell}$. First we start by stating two immediate observations regarding the assignment probabilities of labels to vertices by the basic building blocks (a full proof is deferred to a full version of the paper).

▶ **Lemma 9.** *For any $v \in V$ and $\ell \in K$ the following two claims hold:*
1. $\Pr[\ell \in f_{ST}(v)] = h(x_{v,\ell})$.
2. $\Pr[f_{KT}(v) = \{\ell\}] = \frac{x_{v,\ell}}{||\mathbf{x}_v||_1}$.

The following corollary states the probability that label ℓ is assigned to vertex v by the Union Algorithm (Algorithm 5), and is used to bound the total labeling cost of Algorithm 5. Its proof is deferred to a full version of the paper.

▶ **Corollary 10.** *For any $v \in V$ and $\ell \in K$,* $\Pr[\ell \in f(v)] = h(x_{v,\ell}) + \frac{x_{v,\ell}}{||\mathbf{x}_v||_1} - h(x_{v,\ell}) \cdot \frac{x_{v,\ell}}{||\mathbf{x}_v||_1}$.

We focus now on the separation cost of the Union Algorithm (Algorithm 5). Note that the expected separation cost of the Union Algorithm (Algorithm 5) equals:

$$\sum_{(u,v)\in E} s(u,v) \cdot \mathbb{E}\left[\left|\left|\mathbf{1}_{f(u)} - \mathbf{1}_{f(v)}\right|\right|_1\right] \ . \tag{3}$$

The next lemma provides all the ingredients required for bounding the expected separation cost (3), its proof is deferred to a full version of the paper.

▶ **Lemma 11.** *For any $u, v \in V$ such that $\mathbf{x}_u = (x_{u,1}, x_{u,2}, \ldots, x_{u,k})$ and $\mathbf{x}_v = (x_{u,1} + \varepsilon, x_{u,2}, \ldots, x_{u,k})$, the following hold:*
1. $\Pr[1 \in f(u), 1 \notin f(v)] = 0$.
2. $\Pr[1 \notin f(u), 1 \in f(v)] = \varepsilon \cdot \left(1 - \frac{x_{u,1}}{||\mathbf{x}_u||_1}\right) \cdot \left(\frac{h(x_{u,1}+\varepsilon) - h(x_{u,1})}{\varepsilon} + \frac{1 - h(x_{u,1}+\varepsilon)}{||\mathbf{x}_u|| + \varepsilon}\right)$.
3. $\Pr[\ell \in f(u), \ell \notin f(v)] = \varepsilon \cdot \frac{x_{u,\ell}(1 - h(x_{u,\ell}))}{||\mathbf{x}_u||_1 (||\mathbf{x}_u||_1 + \varepsilon)}$ *for every $\ell \neq 1$.*
4. $\Pr[\ell \notin f(u), \ell \in f(v)] = 0$ *for every $\ell \neq 1$.*

In order to bound the expected separation cost of an edge (u,v), as given by (3), we employ a *global* charging argument. Typically, if *local* charging works it is the case that the part of (3) that corresponds to a fixed label ℓ, *i.e.*, $\mathbb{E}\left[\mathbf{1}_{\{\ell \in f(u) \wedge \ell \notin f(v)\}} + \mathbf{1}_{\{\ell \notin f(u) \wedge \ell \in f(v)\}}\right]$ could be upper bounded by $\alpha \cdot z_{u,v,\ell}$ for some constant $\alpha > 0$. Unfortunately, this is not the case as can be seen from Lemma 11. Edge (u,v) satisfies Observation 5, *i.e.*, $\mathbf{x}_u$ and $\mathbf{x}_v$ differ only coordinate 1, and thus without loss of generality $z_{u,v,\ell} = 0$ for all $\ell \neq 1$. However, for example, case (3) of Lemma 11 implies that u and v have a non-zero probability of disagreeing on any label $\ell \neq 1$. Thus, a local charging argument as described above fails and we must resort to a global argument that sums over all possible labels ℓ. The following corollary provides exactly such a global guarantee, and its proof is deferred to a full version of the paper.

▶ **Corollary 12.** *Let* $\mathbf{x}_u = (x_{u,1}, x_{u,2}, \ldots, x_{u,k})$ *and* $\mathbf{x}_v = (x_{u,1} + \varepsilon, x_{u,2}, \ldots, x_{u,k})$ *for an edge* $(u, v) \in E$*. Then,*

$$\mathbb{E}\left[\left|\left|\mathbf{1}_{f(u)} - \mathbf{1}_{f(v)}\right|\right|_1\right] \leq \left(\frac{h(x_{u,1}+\varepsilon) - h(x_{u,1})}{\varepsilon} + \frac{2 - h(x_{u,1}+\varepsilon)}{||\mathbf{x}_u||_1 + \varepsilon}\right) \cdot \left(1 - \frac{x_{u,1}}{||\mathbf{x}_u||_1}\right) \cdot d_{u,v} \ .$$

Let us not focus on how to choose the distortion h. Given a specific choice of a distortion function $h : [0,1] \to [0,1]$, Corollaries 10 and 12 determine the approximation guarantee. Specifically, Corollary 10 determines the loss with respect to the labeling cost, and Corollary 12 determines the loss with respect to the separation cost.

The most natural distortion function is the identity, i.e., $h(x) = x$. The next theorem shows that this choice of h yields a 3-approximation for (Unbounded-MML).

▶ **Theorem 13.** *The Union Algorithm provides an approximation of* 3 *when* $h(x) = x$*.*

Proof. First, consider the labeling costs. Corollary 10, along with the fact that $||\mathbf{x}_v||_1 \geq 1$, imply:

$$\begin{cases} \Pr\left[\ell \in f(v)\right] = x_{v,\ell} + \frac{x_{v,\ell}}{||\mathbf{x}_v||_1} - \frac{x_{v,\ell}^2}{||\mathbf{x}_v||_1} \leq 2x_{v,\ell} \\ \Pr\left[\ell \notin f(v)\right] = (1 - x_{v,\ell}) \cdot \left(1 - \frac{x_{v,\ell}}{||\mathbf{x}_v||_1}\right) \leq 1 - x_{v,\ell} \end{cases}$$

Hence, the labeling costs incur a loss of at most a factor of 2. Second, consider the separation costs. Let $(u, v) \in E$ and assume without loss of generality that $\mathbf{x}_u = (x_{u,1}, x_{u,2}, \ldots, x_{u,k})$ and $\mathbf{x}_v = (x_{u,1} + \varepsilon, x_{u,2}, \ldots, x_{u,k})$. Corollary 12, along with the fact that $||\mathbf{x}_v||_1 \geq 1$, imply:

$$\mathbb{E}\left[\left|\left|\mathbf{1}_{f(u)} - \mathbf{1}_{f(v)}\right|\right|_1\right] \leq \left(\frac{\varepsilon}{\varepsilon} + \frac{2 - x_{u,1} - \varepsilon}{||\mathbf{x}_u||_1 + \varepsilon}\right)\left(1 - \frac{x_{u,1}}{||\mathbf{x}_u||_1}\right) d_{u,v} \leq 3d_{u,v} \ .$$

Thus, the separation costs incur a loss of at most a factor of 3, concluding the proof. ◀

We prove that choosing a quadratic distortion, i.e., $h(x) = x^2$, provides a tight approximation of 2 for (Unbounded-MML). We are now ready to prove Theorem 3.

Proof (of Theorem 3). For simplicity we prove the theorem in two phases. In the first phase we show that the quadratic distortion provides an approximation of $(2 + \varepsilon)$. In the second phase we show that, assuming $\varepsilon \leq (8k^4)^{-1}$, the approximation is in fact 2. This concludes the proof since ε can be chosen to be arbitrarily small.

Let us focus on the first phase. When considering the labeling costs, Corollary 10, along with the facts that $||\mathbf{x}_v||_1 \geq 1$ and $0 \leq x_{v,\ell} \leq 1$, imply:

$$\begin{cases} \Pr\left[\ell \in f(v)\right] = x_{v,\ell}^2 + \frac{x_{v,\ell}}{||\mathbf{x}_v||_1} - \frac{x_{v,\ell}^3}{||\mathbf{x}_v||_1} \leq x_{v,\ell} \cdot (1 - x_{v,\ell}) \leq 2x_{v,\ell} \\ \Pr\left[\ell \notin f(v)\right] = \left(1 - x_{v,\ell}^2\right) \cdot \left(1 - \frac{x_{v,\ell}}{||\mathbf{x}_v||_1}\right) \leq (1 - x_{v,\ell}^2) \leq 2(1 - x_{v,\ell}) \end{cases}$$

Hence, the labeling costs incur a loss of at most 2 in the approximation.

When considering the separation costs, let $(u, v) \in E$ and assume without loss of generality that $\mathbf{x}_u = (x_{u,1}, x_{u,2}, \ldots, x_{u,k})$ and $\mathbf{x}_v = (x_{u,1} + \varepsilon, x_{u,2}, \ldots, x_{u,k})$. Corollary (12) implies:

$$\begin{aligned} &\mathbb{E}\left[\sum_{\ell \in K}\left(\mathbf{1}_{\{\ell \in f(u) \wedge \ell \notin f(v)\}} + \mathbf{1}_{\{\ell \notin f(u) \wedge \ell \in f(v)\}}\right)\right] \leq \\ &\leq \left(\frac{(x_{u,1}+\varepsilon)^2 - x_{u,1}^2}{\varepsilon} + \frac{2 - (x_{u,1}+\varepsilon)^2}{||\mathbf{x}_u||_1 + \varepsilon}\right)\left(1 - \frac{x_{u,1}}{||\mathbf{x}_u||_1}\right) d_{u,v} \\ &= \left[\left(2x_{u,1} + \frac{2 - x_{u,1}^2}{||\mathbf{x}_u||_1 + \varepsilon}\right)\left(1 - \frac{x_{u,1}}{||\mathbf{x}_u||_1}\right) + \varepsilon\left(\frac{||\mathbf{x}_u||_1 - 2x_{u,1}}{|\mathbf{x}_u||_1 + \varepsilon}\right)\left(1 - \frac{x_{u,1}}{||\mathbf{x}_u||_1}\right)\right] d_{u,v} \ . \quad (4) \end{aligned}$$

Define the following function $L(z,t) : [0,1] \times [1,\infty) \to \mathbb{R}^+$, $L(z,t) \triangleq \left(2z + \frac{2-z^2}{t}\right) \cdot \left(1 - \frac{z}{t}\right)$. Clearly the maximum value of L upper bounds the left term of (4), when plugging $z = x_{u,1}$ and $t = ||\mathbf{x}_u||_1$. One can verify that $\max_{0\le z\le 1} \max_{t\ge 1} \{L(z,t)\} \le 2$ (details are deferred to a full version of the paper). Note that the right term of (4) is at most ε, hence the expected separation cost is at most $(2+\varepsilon)d_{u,v}$. This concludes the first phase. The proof of the second phase is deferred to a full version of the paper. ◀

The proof of Theorem 4 is deferred to a full version of the paper.

References

1 A. A. Ageev and M. I. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8:307–328, 2004.

2 Arash Asadpour, Michel X. Goemans, Aleksander Mądry, Shayan Oveis Gharan, and Amin Saberi. An O($\log n/\log\log n$)-approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'10, pages 379–389, 2010.

3 Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM J. Comput.*, 39(7):2970–2989, 2010.

4 Zafer Barutçuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

5 Hendrik Blockeel, Leander Schietgat, Jan Struyf, Saso Dzeroski, and Amanda Clare. Decision trees for hierarchical multilabel classification: A case study in functional genomics. In *PKDD*, pages 18–29, 2006.

6 Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS'99, pages 450–457, 1999.

7 Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

8 Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *CVPR*, pages 648–655, 1998.

9 Niv Buchbinder, Joseph (Seffi) Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway cut problem. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC'13, pages 535–544, 2013.

10 Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrak. Maximizing a submodular set function subject to a matroid constraint. *SIAM Journal on Computing*, 2011.

11 Gruia Călinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. In *SODA'01*, pages 8–16, 2001.

12 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 575–584, 2010.

13 R Chellappa and A. Jain. *Markov Random Fields: Theory and Applications*. Academic Press, 1993.

14 Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2002.

15 Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for multiway cut. *J. Comput. Syst. Sci.*, 60(3):564–574, 2000.

16 William H. Cunningham and Lawrence Tang. Optimal 3-terminal cuts and linear programming. In *IPCO'99*, pages 114–125, 1999.

17 E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

18 Richard C. Dubes and Anil K. Jain. Random field models in image analysis. *Journal of Applied Statistics*, 16(2):131–164, 2006.

19 Jittat Fakcharoenphol, Chris Harrelson, Satish Rao, and Kunal Talwar. An improved approximation algorithm for the 0-extension problem. In *SODA'03*, pages 257–265, 2003.

20 Evelyn Fix and Joseph L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

21 Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

22 Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360, 2006.

23 David R. Karger, Philip N. Klein, Clifford Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Math. Oper. Res.*, 29(3):436–461, 2004.

24 Hisashi Kashima, Satoshi Oyama, Yoshihiro Yamanishi, and Koji Tsuda. On pairwise kernels: An efficient alternative and generalization analysis. In *Advances in Knowledge Discovery and Data Mining*, pages 1030–1037. Springer, 2009.

25 Jon M. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, 2002.

26 V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. A unified approach to scheduling on unrelated parallel machines. *J. ACM*, 56(5):28:1–28:31, 2009.

27 Stan Z. Li. *Markov random field modeling in computer vision*. Computer science workbench. Springer, 1995.

28 Rajsekar Manokaran, Joseph (Seffi) Naor, Prasad Raghavendra, and Roy Schwartz. Sdp gaps and ugc hardness for multiway cut, 0-extension, and metric labeling. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC'08, pages 11–20, 2008.

29 Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):380–393, 1997.

30 Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of ACMMM'07*, pages 17–26, 2007.

31 J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Elsevier, 2014.

32 Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

33 Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

34 Mohammad S Sorower. A literature survey on algorithms for multi-label learning. Technical report, 2010.

35 A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 588–597, 2001.

36 Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

37 Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *In Data Mining and Knowledge Discovery Handbook*, pages 667–685, 2010.

38 Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. In *NIPS*, pages 721–728, 2002.

39 Adriano Veloso, Wagner Meira Jr., Marcos André Gonçalves, and Mohammed Javeed Zaki. Multi-label lazy associative classification. In *PKDD*, pages 605–612, 2007.

40 Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.

41 Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40:2007, 2007.

42 Zhi-Hua Zhou and Min-Ling Zhang. Multi-instance multi-label learning with application to scene classification. In *Proceedings of NIPS*, pages 1609–1616, 2006.

When the Optimum is also Blind: a New Perspective on Universal Optimization*†

Marek Adamczyk[1], Fabrizio Grandoni[2], Stefano Leonardi[3], and Michał Włodarczyk[4]

1 **University of Bremen, Bremen, Germany**
m.adamczyk@uni-bremen.de
2 **IDSIA, USI-SUPSI, Lugano, Switzerland**
fabrizio@idsia.ch
3 **Sapienza University of Rome, Rome, Italy**
leonardi@dis.uniroma1.it
4 **University of Warsaw, Warsaw, Poland**
m.wlodarczyk@mimuw.edu.pl

Abstract

Consider the following variant of the set cover problem. We are given a universe $U = \{1, ..., n\}$ and a collection of subsets $\mathcal{C} = \{S_1, ..., S_m\}$ where $S_i \subseteq U$. For every element $u \in U$ we need to find a set $\phi(u) \in \mathcal{C}$ such that $u \in \phi(u)$. Once we construct and fix the mapping $\phi : U \mapsto \mathcal{C}$ a subset $X \subseteq U$ of the universe is revealed, and we need to cover all elements from X with exactly $\phi(X) := \bigcup_{u \in X} \phi(u)$. The goal is to find a mapping such that the cover $\phi(X)$ is as cheap as possible.

This is an example of a universal problem where the solution has to be created before the actual instance to deal with is revealed. Such problems appear naturally in some settings when we need to optimize under uncertainty and it may be actually too expensive to begin finding a good solution once the input starts being revealed. A rich body of work was devoted to investigate such problems under the regime of worst case analysis, i.e., when we measure how good the solution is by looking at the worst-case ratio: universal solution for a given instance vs optimum solution for the same instance.

As the universal solution is significantly more constrained, it is typical that such a worst-case ratio is actually quite big. One way to give a viewpoint on the problem that would be less vulnerable to such extreme worst-cases is to assume that the instance, for which we will have to create a solution, will be drawn randomly from some probability distribution. In this case one wants to minimize the expected value of the ratio: universal solution vs optimum solution. Here the bounds obtained are indeed smaller than when we compare to the worst-case ratio.

But even in this case we still compare apples to oranges as no universal solution is able to construct the optimum solution for every possible instance. What if we would compare our approximate universal solution against an optimal universal solution that obeys the same rules as we do? We show that under this viewpoint, but still in the stochastic variant, we can indeed obtain better bounds than in the expected ratio model. For example, for the set cover problem we obtain H_n approximation which matches the approximation ratio from the classic deterministic offline setup. Moreover, we show this for all possible probability distributions over U that have a polynomially large carrier, while all previous results pertained to a model in which elements were sampled independently. Our result is based on rounding a proper configuration IP that captures the optimal universal solution, and using tools from submodular optimization.

The same basic approach leads to improved approximation algorithms for other related problems, including Vertex Cover, Edge Cover, Directed Steiner Tree, Multicut, and Facility Location.

* A full version of the paper is available at http://arxiv.org/abs/1707.01702.

† The second author was partially supported by the ERC Starting Grant NEWNET 279352 and the SNSF Grant APPROXNET 200021_159697/1. The fourth author was partially supported by the National Science Centre of Poland Grant UMO-2016/21/N/ST6/00968.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 35; pp. 35:1–35:15

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

2012 ACM Subject Classification F. Theory of Computation, G.1.6 Optimization

Keywords and phrases approximation algorithms, stochastic optimization, submodularity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.35

1 Introduction

In a typical online problem part of the input is revealed gradually to an algorithm, which has to react to each new piece of the input by making irrevocable choices. In an online covering problem the online input consists of a sequence of *requests*, which have to be satisfied by the algorithm by buying *items* at minimum total cost.

Some online applications have severe resource constraints, typically in terms of time and/or computational power. Hence even making an online (non-trivial) choice might be too costly. In these settings it makes sense to consider universal algorithms. Roughly speaking, the goal of these algorithms is to pre-compute a reaction to each possible input, so that the online choice can then be made very quickly (say, looking at some pre-computed table). Since the adversary has a lot of power in the universal setting, typically one assumes a stochastic input. In particular, the input is sampled according to some probability distribution π, which is either given in input or that can be sampled multiple times at polynomial cost per sample (*oracle model*).

The most relevant prior work for this paper is arguably due to Grandoni et al. [21] (conference version in [20]). The authors consider the universal stochastic version of some classical NP-hard covering problems such as set cover, non-metric facility location, multicut etc. They provide polynomial-time approximation algorithms for those problems in the *independent activation model*, where each request u is independently sampled with some known probability p_u. Crucially, in their work the approximation ratio is obtained by comparing the expected cost of the approximate solution with the expected cost of the optimal offline solution (that knows the future sampled input). For example, in the set cover case they present a polynomial-time algorithm that computes a mapping of expected cost at most $O(\log(nm))\,\mathbb{E}[OPT_{\text{off}}(X)]$, where the expectation is taken over the sampling of X according to π and $OPT_{\text{off}}(X)$ is the minimum (offline) cost of a set cover of X. Here n is size of the universe and m the number of subsets. For $m \gg n$ this ratio becomes $O\left(\frac{\log m}{\log\log m}\right)$ and is tight. They also consider the universal (non-metric) facility location problem in the independent activation model, and provide a $O(\log n)$ approximation (in the above sense), where n is the total number of clients and facilities. We remark that their method seems not to lead to any improved approximation factor in the metric version of the problem. We finally mention their $O(\log^2 n)$ approximation for universal multicut in the independent activation model, where n is the number of nodes in the graph.

1.1 Our Results and Techniques

Comparing with the offline optimum as in [21] might be too pessimistic. And often when we need to optimize under uncertainty we cannot really find a better benchmark, flagship example of it would be online problems. However, stochastic two-stage [29, 42] and stochastic adaptive [38, 13, 12, 23] problems have proven that one can actually compare an approximate solution with an optimum algorithm that is not omnipotent but obeys the same rules of the model as the approximate one. This inspired us to ask the following question:

Is it also possible in a stochastic universal problem to compare our algorithm with an optimum solution that is restricted by the model in the same way as we are?

In this paper we show that we can do this indeed. In this way we manage to obtain tighter approximation ratios – which of course are compared to a weaker benchmark, but this benchmark itself can be interpreted as more fair and meaningful – and it also allows us to approach more general problems.

1.1.1 Universal Stochastic Set Cover

We shall describe carefully the Universal Stochastic Set Cover problem in this section so that we will fully present the model. For the remaining problems their full statements will appear in appropriate sections.

In the Universal Stochastic Set Cover problem we are given in input a universe $U = \{1, 2, ..., n\}$, and a collection $\mathcal{C} \subseteq 2^U$ of m subsets $S \subseteq U$, each one with an associated cost $c(S)$. We need to a priori map each element $u \in U$ into some set $\phi(u) \in \mathcal{C}$. Then a subset $X \subseteq U$ is sampled according to some probability distribution π (either given in input as a set of scenarios or only by accessing an oracle), and we have to buy the sets $\phi(X) = \bigcup_{u \in X} \phi(u)$ as the cover of X. Our goal is to minimize the expected value of the total cost, i.e., $\mathbb{E}_{X\sim\pi}[c(\phi(X))] = \mathbb{E}_{X\sim\pi}\left[\sum_{S\in\phi(X)} c(S)\right]$. One of the most important aspects in our model is that we do not compare ourselves against the expected value of an optimum offline solution for a given scenario, that is, not against $\mathbb{E}_{X\sim\pi}[OPT_{\text{off}}(X)]$, but what we compare ourselves with is

$$\min_{\phi:\forall_{u\in U} u\in\phi(u)} \mathbb{E}_{X\sim\pi}\left[\sum_{S\in\phi(X)} c(S)\right]$$

which is the expected outcome of an *optimal universal mapping*.

The results depend on the probability distribution with which we deal in an instance of the problem. Possible probability distributions are:

- **Scenario model:** Here we are given in input all the sets $X_1, ..., X_N \subseteq U$ such that $\mathbb{P}_{X\sim\pi}[X = X_i] > 0$ with the associated probability. This model allows for explicit use of all the scenarios in the computations. For the Universal Stochastic Set Cover we obtain $O(\log n)$-approximation in this case even in the weighted case.
- **Oracle model:** This is the most general model. We have a black-box access to an oracle Π from which we can sample a scenario from distribution π. We assume that taking a sample requires polynomial time. In this model we can find an $O(\log n)$-approximation for Universal Stochastic Set Cover in polynomial time only for the unweighted case; in the weighted case we achieve the same approximation factor in pseudo-polynomial time depending on $\max_{S\in\mathcal{C}} c(S)$.
- **Independent activation model:** In this model we assume that every element $u \in U$ is independently sampled with some given probability p_u. This model does not capture correlations of elements, and therefore sometimes it is not fully realistic. Though it cannot be represented by a polynomial number of scenarios, its nice properties allow one to develop good approximation algorithms for several problems. In this setting we are able to approximate Universal Stochastic Set Cover within a factor $O(\log n)$ in polynomial time even in the weighted case.

Our results are obtained by defining a proper configuration LP (with an exponential number of variables) that captures the optimal mapping. We are able to solve this LP via the ellipsoid

method using a separation oracle. Somehow interestingly, our separation oracle has to solve a submodular minimization problem. Then we can round the fractional solution in a standard way.

1.1.2 Overview of the results

The robustness of our framework allows us to address universal extensions of several covering problems. After expressing the goal as a true approximation task, we can adapt tools from the rich theory of approximation algorithms.

Here we give an overview of our results. Detailed statements of the theorems appear in appropriate sections.

Scenario model

In this setting, we are able to construct an LP-based polynomial-time $O(\log n)$-approximation to the universal stochastic version of Set Cover (Theorem 6), which generalizes to Non-Metric Facility Location and Constrained Set Multicover. In fact, the latter algorithm achieves an approximation guarantee of exactly H_n. Different rounding procedure leads to a 2-approximation for Universal Stochastic Vertex Cover. All these approximation ratios match the best guarantees obtained in the deterministic world. What is more, the exact polynomial time algorithm for Edge Cover extends to the scenario model. We also present an $O(\sqrt{k})$-approximation for Universal Stochastic Directed Steiner Tree on acyclic graphs, where k is the number of terminals. Except the Set Cover problem, proof of all of these Theorems will appear in the full version of the paper.

Independent activation model

In this setting, we are able to obtain several results in flavour of the $O(1)$-approximation for the Maybecast problem by Karger and Minkoff [33]. We present a 6.33-approximation for Universal Stochastic Metric Facility Location (Theorem 8) and an $O(\log n)$-approximation for Universal Stochastic Multicut (Theorem 12). As an intermediate result, we obtain a 4.75-approximation for Universal Stochastic Multicut on trees.

Oracle model

We can generalize most of our results for the scenario model to this setting, with the restriction that in the weighted case we get a pseudo-polynomial running time. This will be discussed in the full version.

1.2 Related work

Other universal-like problems have been addressed in the literature. For instance, in the universal TSP problem one computes a permutation of the nodes that is then used to visit a given subset of nodes. This problem has been studied both in the worst-case scenario for the Euclidean plane [39, 6] and general metrics [32, 22, 27], as well as in the average-case [31, 7, 43, 19, 45]. (For the related problem of universal Steiner tree, see [33, 32, 22, 19].) Jia et al. [32] introduced the universal set cover and universal facility location problems, and studied them in the worst-case: they show that the adversary is very powerful in such models, and give nearly-matching $\Omega(\sqrt{n})$ and $O(\sqrt{n \log n})$ bounds on the competitive factor. These problems have been later studied by Grandoni et al. in the independent activation model [21], as already mentioned before.

A somewhat related topic is *oblivious routing* [40, 28, 8] (see, e.g., [47, 49] for special cases). A tight logarithmic competitive result as well as a polynomial-time algorithm to compute the best routing is known in the worst case for undirected graphs [5, 41]. For *oblivious routing on directed graphs* in the worst case the lower bound of $\Omega(\sqrt{n})$ [5] nearly matches upper bounds in [24] but for the average case. The authors of [25] give an $O(\log^2 n)$-competitive oblivious routing algorithm when demands are chosen randomly from a known demand-distribution; they also use "demand-dependent" routings and show that these are necessary.

Another closely related notion is the one of *online problems*. These problems have a long history (see, e.g., [9, 16]), and there have been many attempts to relax the strict worst-case notion of competitive analysis: see, e.g., [14, 1, 19] and the references therein. Online problems with stochastic inputs (either i.i.d. draws from some distribution, or inputs arriving in random order) have been studied, e.g., in the context of optimization problems [36, 37, 19, 4], secretary problems [18], mechanism design [26], and matching problems in Ad-auctions [35].

Alon et al. [2] gave the first online algorithm for set cover with a competitive ratio of $O(\log m \log n)$; they used an elegant primal-dual-style approach that has subsequently found many applications (e.g., [3, 10]). This ratio is the best possible under complexity-theoretic assumptions [34]; even unconditionally, no deterministic online algorithm can do much better than this [2]. Online versions of *metric* facility location are studied in both the worst case [36, 17], the average case [19], as well as in the stronger *random permutation model* [36], where the adversary chooses a set of clients unknown to the algorithm, and the clients are presented to us in a random order. It is easy to show that for our problems, the random permutation model (and hence any model where elements are drawn from an *unknown* distribution) are as hard as the worst case.

One can of course consider the (offline) stochastic version of optimization problems. For example, k-stage stochastic set cover is studied in [29, 44], with an improved approximation factor (independent from k) later given in [46].

The result for the Oracle model are based on the Sample Average Approximation approach, see [11] for the application most relevant to our work.

As mentioned before, in two-stage stochastic problems [42, 29] and stochastic adaptive problems [13, 38, 12, 23] it is possible to compare our algorithms with an optimum algorithm which is equally constrained in the model as our problem, and this is what shed a light on the possibility of obtaining results in the same spirit for the universal stochastic optimization.

In two recent papers [48, 15] authors looked at universal optimization over scenarios, but compared against the average offline optimum, and not the optimum universal solution as we do. All the properties of submodular functions used in our work can be found here [50].

2 Preliminaries

Here we give some basic definitions and properties. Given a universe U, we call a function $f : 2^U \to \mathbb{R}$ submodular if $f(A)+f(B) \geq f(A\cap B)+f(A\cup B)$ for each pair of sets $A, B \subseteq U$. Function f is monotone if $f(B) \geq f(A)$ for $A \subseteq B$. When considering a submodular function, we assume that it is implicitly given in the form of an oracle that can be queried on a specific $A \subseteq U$ and returns the value $f(A)$ in constant time.

▶ **Theorem 1** (Iwata et al. [30]). *There is an algorithm to minimize a given submodular function $f : 2^U \to \mathbb{N}$ in polynomial time in $|U|$ and in the number of bits needed to encode the largest value of f.*

Let us introduce a function $g_\pi : 2^U \to \mathbb{R}^+$, $g_\pi(A) = \mathbb{P}[A \cap X \neq \emptyset]$ where X is drawn from the distribution π. Our framework exploits crucially the fact that g_π is a submodular function.

▶ **Lemma 2.** *Function g_π is submodular and monotone.*

Proof. Observe that $g_\pi(A) = \sum_{X \subseteq U} \pi(X) \cdot \mathbf{1}[A \cap X \neq \emptyset]$. Function $A \to \mathbf{1}[A \cap X \neq \emptyset]$ is submodular and a combination of such functions with positive coefficients is submodular. The monotonicity holds trivially by definition. ◀

3 Universal Set Cover

In this section we present our approximation algorithm for universal set cover. We start by considering the case that π is given in input, in the form of a set of N scenarios $X_1, \ldots, X_N$, where scenario $X_i \subseteq U$ is sampled with some given probability $p_i = \pi(X_i)$. Then we (partially) extend our approach to the oracle model. Recall that $n = |U|$ is the number of elements in the universe.

Consider the scenario case. Recall that, for $B \subseteq U$, $g_\pi(B) = \mathbb{P}[B \cap X \neq \emptyset]$, where the probability is taken over $X \subseteq U$ sampled from π (namely, one of the scenarios A_i in this case). As mentioned before, g_π is a submodular function over the universe U.

We start by expressing our problem as the following integer program (IP).

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{C}} c(S) \sum_{B \subseteq S} y_B^S \cdot g_\pi(B) && \text{(IP-SC)} \\ \text{s.t.} \quad & \sum_{B \ni u} \sum_{S \supseteq B} y_B^S \geq 1 && \forall_{u \in U} \\ & y_B^S \in \{0,1\} && \forall_{S \in \mathcal{C}} \forall_{B \subseteq S}. \end{aligned} \tag{1}$$

We obtain a linear relaxation (LP-SC) of (IP-SC) by replacing the integrality constraints (1) with $y_B^S \geq 0$.

▶ **Lemma 3.** *Integer program (IP-SC) is equivalent to Universal Stochastic Set Cover.*

Proof. It is easy to translate a mapping $\phi : U \to \mathcal{C}$ into some feasible solution to (LP-SC). All variables are zeros by default and for each $S \in \mathcal{C}$ such that $\phi^{-1}(S)$ is non-empty, we set $y^S_{\phi^{-1}(S)} = 1$. Note that always $\phi^{-1}(S) \subseteq S$. In that setting the objective value equals the expected cost of the covering.

Let us fix some feasible solution $\{y_B^S\}_{S \in \mathcal{C},\, B \subseteq S}$. We know that for each $u \in U$ there is some pair (B, S) so that $u \in B$ and $y_B^S = 1$ (we will call it a *covering pair*). As long as there are many covering pairs for some u, we replace one of them with $(B \setminus \{u\}, S)$. The new solution is still feasible and the objective value is no greater as function g_π is monotone. Therefore there exists an optimal solution so that each $u \in U$ admits exactly one covering pair (B_u, S_u). We can define $\phi(u) = S_u$ to obtain a covering with expected cost equal to the value of the objective function. ◀

(LP-SC) has an exponential number of variables: in order to solve it we consider its dual, and provide a separation oracle to solve it. Interestingly, our separation oracle uses submodular minimization.

▶ **Lemma 4.** *(LP-SC) can be solved in polynomial time.*

Proof. We show how to solve the dual of (LP-SC), which is as follows:

$$\begin{aligned} \max \quad & \sum_{u\in U} \alpha_u && \text{(DP-SC)} \\ \text{s.t.} \quad & \sum_{u\in B} \alpha_u \le c(S)\cdot g_\pi(B) && \forall_{S\in\mathcal{C}}\forall_{B\subseteq S} \\ & \alpha_u \ge 0 && \forall_{u\in U}. \end{aligned}$$

Observe that (DP-SC) has a polynomial number of variables and an exponential number of constraints. In order to solve (DP-SC), it is sufficient to provide a (polynomial-time) separation oracle, i.e. a procedure that, given a tentative solution $\{\alpha_u\}_{u\in U}$, either determines that it is feasible or provides a violated constraint.

This reduces to check, for each given $S \in \mathcal{C}$, whether there exists $B \subseteq S$ such that $\sum_{u\in B}\alpha_u > c(S)\cdot g_\pi(B)$. In other terms, we wish to determine whether the minimum of function $h_S(B) := c(S)\cdot g_\pi(B) - \sum_{u\in B}\alpha_u$ is negative. Observe that the value of $h_S(B)$ can be computed in polynomial time for a given B, since π is given in input as a set of scenarios. Note also that h_S is submodular: indeed g_π is submodular, costs $c(S)$ are non-negative by assumption, and $-\sum_{u\in B}\alpha_u$ is linear (hence submodular). Hence we can minimize h_S over $B \subseteq S$ in polynomial time via Theorem 1.[1] ◀

Given the optimal solution to (LP-SC), it is sufficient to round it with the usual randomized rounding algorithm for set cover.

▶ **Lemma 5.** *The optimal solution to (LP-SC) can be rounded to an integer feasible solution while increasing the cost by a factor $O(\log n)$ in expected polynomial time.*

Proof. In the optimal solution, for each $S \in \mathcal{C}$, variables $\{y_B^S\}_{B\subseteq S}$ define a probability distribution. We sample from this distribution (independently for each S) for $q = 2\ln n$ many times. Let $B_1^S, \dots, B_q^S$ be the sets sampled for S: we let $B^S := \cup_i B_i^S$ and tentatively map elements of B^S into S. In case the same element u belongs to $B^{S'}$ and $B^{S''}$ for $S' \neq S''$, we replace $B^{S'}$ with $B^{S'} \setminus \{u\}$ and iterate: this way each element is mapped into exactly one set. The final sets B^S induce our approximate mapping.

We can upper bound the expected cost of the solution by $\sum_{S\in\mathcal{C}} c(S)\sum_{i=1}^q g_\pi(B_i^S)$. Indeed, by the subadditivity of g_π (which is implied by submodularity and non-negativity), one has $g_\pi(B^S) \le \sum_{i=1}^q g_\pi(B_i^S)$. Furthermore, $g_\pi(B^{S'} \setminus \{u\}) \le g_\pi(B^{S'})$ since g_π is monotone. Trivially

$$\begin{aligned} \mathbb{E}\left[\sum_{S\in\mathcal{C}} c(S)\sum_{i=1}^q g_\pi(B_i^S)\right] &= 2\ln n\cdot \mathbb{E}\left[\sum_{S\in\mathcal{C}} c(S) g_\pi(B_1^S)\right] \\ &= 2\ln n\cdot \sum_{S\in\mathcal{C}} c(S)\sum_{B\subseteq S} y_B^S\cdot g_\pi(B^S) \le 2\ln n\cdot OPT. \end{aligned}$$

And from Markov's inequality

$$\mathbb{P}\left[\sum_{S\in\mathcal{C}} c(S)\sum_{i=1}^q g_\pi(B_i^S) > 4\ln n\cdot OPT\right] < \frac{1}{2}.$$

[1] In order to solve the configuration LP, there is an alternative to finding a separation oracle for the dual. We can transform the configuration LP into a optimization program where we need to minimize a sum of Lovasz's extensions [50], which are convex functions, over a convex region. This approach would be possibly more efficient, but we have chosen the one above for a simpler presentation.

The probability that an element $u \in U$ is not covered with a single sampling over y_B^S is

$$\prod_{B \ni u} \prod_{S \supseteq B} (1 - y_B^S) \leq \prod_{B \ni u} \prod_{S \supseteq B} e^{-y_B^S} = e^{-\sum_{B \ni u} \sum_{S \supseteq B} y_B^S} \leq \frac{1}{e}.$$

Therefore, by the independence of the sampling and the union bound, the probability that at least one element is not covered is at most $n \cdot \frac{1}{e^{2 \ln n}} = \frac{1}{n}$.

Altogether this gives a Monte-Carlo algorithm. As usual, this can be turned into a Las-Vegas algorithm with expected polynomial running time by repeating the procedure when some element is not covered or the cost of the solution is greater than $4 \ln n \cdot OPT$. ◀

The following theorem is a straight-forward consequence of Lemmas 4 and 5.

▶ **Theorem 6.** Universal Stochastic Set Cover *in the scenario model admits a polynomial-time $O(\log n)$ approximation algorithm w.r.t. the optimal universal mapping.*

It turns out that not only the randomized rounding technique can be adapted to the universal stochastic model but also the dual fitting technique. This allows us to improve the above result to a purely deterministic algorithm with the approximation ratio exactly equal to H_n. What is more, we are able to consider more general problems where each element u is required to be covered at least $r(u)$ times. Details will appear in the full version.

Our results can be also partially generalized to the oracle model, where the probability distribution is given only by a blackbox oracle. In the unweighted case we are able to give a polynomial time algorithm, while in the weighted case a pseudo-polynomial time one where the complexity depends also on $\max_S c(S)$. Again, details will appear in the full version.

Finally, since in the independent activation model we can compute function g_π in polynomial time, our approach works also in that case.

▶ **Theorem 7.** Universal Stochastic Set Cover *in the independent activation model admits a polynomial-time $O(\log n)$ approximation algorithm w.r.t. the optimal universal mapping.*

4 Metric facility location in the independent activation model

In the stochastic universal variant of the metric uncapacitated facility location (*UniStoch-FL*) problem, we have a set of clients C and a set of facilities F. For each client $c \in C$ and facility $f \in F$, there is a cost $d(c, f)$ paid if c is connected to f; furthermore, there is a cost o_f associated with opening facility $f \in F$. In the universal solution we need to assign every client $c \in C$ to a facility $\phi(c) \in F$. Once a set $X \subseteq U$ is realized we need to open all facilities $f \in \bigcup_{c \in X} \phi(c)$ and connect each $c \in X$ to $\phi(c)$. The goal is to minimize the expected total cost of opening the facilities and connecting clients to facilities:

$$\mathbb{E}_{X \sim \pi} \left[\sum_{f \in \bigcup_{c \in X} \phi(c)} o_f + \sum_{c \in X} d(c, \phi(c)) \right].$$

Just by direct modeling of the above formula with the configuration LP we can see that the following is a relaxation of an integer program that solves the problem:

$$\min \sum_{f \in F} o_f \sum_{B \subseteq C} y_B^f \cdot g_\pi(B) + \sum_{c \in C} \sum_{f \in F} g_\pi(c) \cdot d(c, f) \cdot \sum_{B \subseteq C : c \in B} y_B^f \qquad \text{(CONF-LP-FL)}$$

$$\text{s.t. } \forall_{c \in C} : \sum_{f \in F} \sum_{B \subseteq C : c \in B} y_B^f \geq 1 \qquad \text{and} \qquad \forall_{f \in F} \forall_{B \subseteq C} : y_B^f \geq 0.$$

Let $OPT_{CONF-LP-FL}$ be the optimal solution to the above LP. It holds that $OPT_{CONF-LP-FL}$ is a lower-bound on the expected outcome of an optimal universal solution.

What we are able to present in this section is a constant approximation in the case of independent activation where every client c is independently chosen into the scenario with probability p_c. This restriction is due to the fact that we do not know of a good rounding procedure in the general case, while we can solve the configuration LP even with the arbitrary scenario distribution.

▶ **Theorem 8.** *For* Universal Stochastic Facility Location *in the independent activation model there exists an algorithm with expected outcome cost at most* $6.33 \cdot OPT_{CONF\text{-}LP\text{-}FL}$.

4.1 Transforming the LP

We start off with the following two inequalities for any $S \subseteq C$ (see [33] for proof):

$$\min\left(1, \sum_{t\in S} p_t\right) \geq g_\pi(S) = 1 - \prod_{t\in S}(1-p_t) \geq \left(1-\frac{1}{e}\right)\min\left(1, \sum_{t\in S} p_t\right). \tag{2}$$

Now based on this inequality we shall transform and relax the (CONF-LP-FL) to get one that is a bit easier to tackle. The inequality implies that the solution of the following LP is at most $\frac{e}{e-1}$ bigger than the solution of (CONF-LP-FL):

$$\begin{aligned}
&\min \sum_{f\in F} o_f \sum_{B\subseteq C} y_B^f \cdot \min\left(1, \sum_{c\in B} p_c\right) + \sum_{c\in C}\sum_{f\in F} g_\pi(c)\cdot d(c,f)\cdot \sum_{B\subseteq C: c\in B} y_B^f && (3)\\
&\text{s.t.} \sum_{f\in F}\sum_{B\subseteq C: c\in B} y_B^f \geq 1 && \forall_{c\in C}\\
&\quad y_B^f \geq 0 && \forall_{f\in F}\forall_{B\subseteq C}.
\end{aligned}$$

Let Big be a collection of sets $B \subseteq C$ such that $\sum_{t\in B} p_t > 1$, and let Sml be a collection of sets $B \subseteq C$ such that $\sum_{t\in B} p_t \leq 1$. Define x_c^f to be the extent to which c was assigned to f via sets from Big, and $\bar{x}_c^f$ the extent to which c was assigned to f via sets from Sml, i.e., $x_c^f = \sum_{B\in Big: c\in B} y_B^f$ and $\bar{x}_c^f = \sum_{B\in Sml: c\in B} y_B^f$. Also, for every client c there is an obvious inequality $\sum_{B\in Big} y_B^f \geq \sum_{B\in Big: c\in B} y_B^f = x_c^f$. Now we can lower bound (3):

$$\begin{aligned}
&\sum_{f\in F} o_f \left(\sum_{B\in Big} y_B^f + \sum_{B\in Sml} y_B^f \cdot \left(\sum_{j\in B} p_j\right)\right) + \sum_{c\in C}\sum_{f\in F} p_c\cdot d(c,f)\cdot \sum_{B\subseteq C: c\in B} y_B^f\\
&= \sum_{f\in F} o_f \left(\sum_{B\in Big} y_B^f\right) + \sum_{f\in F}\sum_{c\in C} o_f\cdot p_c \left(\sum_{B\in Sml: c\in B} y_B^f\right) + \sum_{c\in C}\sum_{f\in F} p_c\cdot d(c,f)\cdot \sum_{B\subseteq C: c\in B} y_B^f\\
&\geq \sum_{f\in F} o_f \cdot \max_{c\in C}\left(x_c^f\right) + \sum_{f\in F}\sum_{c\in C} o_f\cdot p_c\cdot \bar{x}_c^f + \sum_{c\in C}\sum_{f\in F} p_c\cdot d(c,f)\cdot\left(x_c^f + \bar{x}_c^f\right).
\end{aligned}$$

▶ **Lemma 9.** *Let* OPT_{LP-FL} *be the optimum solution of the following LP:*

$$\begin{aligned}
&\min \sum_{f\in F} o_f \cdot \max_{c\in C}\left(x_c^f\right) + \sum_{f\in F} o_f \cdot \sum_{c\in C} p_c\cdot \bar{x}_c^f + \sum_{c\in C}\sum_{f\in F} p_c\cdot d(c,f)\cdot\left(x_c^f + \bar{x}_c^f\right) && \text{(LP-FL)}\\
&\text{s.t. } \forall_{c\in C}: \sum_{f\in F} x_c^f + \bar{x}_c^f \geq 1 \qquad \text{and} \qquad \forall_{f\in F}\forall_{c\in C}: x_c^f, \bar{x}_c^f \geq 0.
\end{aligned}$$

It holds that $OPT_{LP-FL} \leq \frac{e}{e-1}\cdot OPT_{CONF-LP-FL}$.

4.2 The rounding procedure

LP-FL has a polynomial number of variables and constraints, and so it can be solved in polynomial time; denote by $\left(x_c^f, \bar{x}_c^f\right)_{c\in C, f\in F}$ the optimal solution. Once we solve it we split the clients into two groups:

$$C_{\text{big}} := \left\{ c \in C \,\middle|\, \sum_{f\in F} x_c^f \geq \frac{3}{4} \right\} \text{ and } C_{\text{sml}} = \left\{ c \in C \,\middle|\, \sum_{f\in F} \bar{x}_c^f > \frac{1}{4} \right\}.$$

Dealing with clients from C_{big}

From the definition we get that $\left(\frac{4}{3}x_c^f\right)_{c\in C_{\text{big}}, f\in F}$ is a feasible solution to the following problem, which is a variant of the deterministic facility location problem where the price for the distance for client $c \in C$ is proportional to the factor p_c:

$$\begin{aligned} \min \; & \sum_{f\in F} o_f \cdot \max_{c\in C_{\text{big}}} \left(z_c^f\right) + \sum_{c\in C}\sum_{f\in F} p_c \cdot d\left(c,f\right) \cdot z_c^f \\ \text{s.t.} \; & \forall_{c\in C_{\text{big}}} : \sum_{f\in F} z_c^f \geq 1 \qquad \text{and} \qquad \forall_{f\in F}\forall_{c\in C_{\text{big}}} : z_c^f \geq 0. \end{aligned}$$

For this problem there exists a 3-approximation algorithm based on primal-dual method (the proof is a simple exercise and will appear in the full version of the paper); let $\left(z_c^f\right)_{c\in C, f\in F}$ be such an integral solution. Thus we have

$$\begin{aligned} & \sum_{f\in F} o_f \cdot \max_{c\in C_{\text{big}}} \left(z_c^f\right) + \sum_{c\in C_{\text{big}}}\sum_{f\in F} p_c \cdot d\left(c,f\right) \cdot z_c^f \\ \leq\; & 3 \cdot \sum_{f\in F} o_f \cdot \max_{c\in C_{\text{big}}} \left(\frac{4}{3}x_c^f\right) + 3 \cdot \sum_{c\in C_{\text{big}}}\sum_{f\in F} p_c \cdot d\left(c,f\right) \cdot \frac{4}{3}x_c^f \\ \leq\; & 4 \cdot \sum_{f\in F} o_f \cdot \max_{c\in C} \left(x_c^f\right) + 4 \cdot \sum_{c\in C}\sum_{f\in F} p_c \cdot d\left(c,f\right) \cdot x_c^f. \end{aligned} \tag{4}$$

Dealing with clients from C_{sml}

Now the $\left(4 \cdot \bar{x}_c^f\right)_{c\in C_{\text{sml}}, f\in F}$ vector is a feasible solution to the following LP:

$$\begin{aligned} \min \; & \sum_{c\in C_{\text{sml}}}\sum_{f\in F} p_c \left(o_f + d\left(c,f\right)\right) \cdot z_c^f \\ \text{s.t.} \; & \forall_{c\in C_{\text{sml}}} : \sum_{f\in F} z_c^f \geq 1 \qquad \text{and} \qquad \forall_{f\in F}\forall_{c\in C_{\text{sml}}} : z_c^f \geq 0. \end{aligned}$$

We can find an integral solution to this LP easily: just assign every client $c \in C_{\text{sml}}$ to the facility f that minimizes $o_f + d\left(c,f\right)$. If $\left(z_c^f\right)_{c\in C_{\text{sml}}, f\in F}$ is such an integral solution, then we have that

$$\sum_{c\in C_{\text{sml}}}\sum_{f\in F} p_c \left(o_f + d\left(c,f\right)\right) \cdot z_c^f \leq \sum_{c\in C_{\text{sml}}}\sum_{f\in F} p_c \left(o_f + d\left(c,f\right)\right) \cdot 4\bar{x}_c^f. \tag{5}$$

Combining the two solutions

In this way the solution composed from two integer vectors $\left(z_c^f\right)_{c\in C_{\text{big}},f\in F}$ and $\left(z_c^f\right)_{c\in C_{\text{sml}},f\in F}$ is a feasible solution to the (LP-FL), and by combining (4) and (5) its value is

$$\begin{aligned}
&\sum_{f\in F} o_f \cdot \max_{c\in C_{\text{big}}} \left(z_c^f\right) + \sum_{f\in F}\sum_{c\in C_{\text{sml}}} o_f \cdot p_c \cdot z_c^f + \sum_{c\in C}\sum_{f\in F} p_c \cdot d\left(c,f\right)\cdot z_c^f \\
\leq\ & 4\cdot \sum_{f\in F} o_f \cdot \max_{c\in C} \left(x_c^f\right) + \sum_{f\in F}\sum_{c\in C} o_f \cdot p_c \cdot 4\bar{x}_c^f + \sum_{c\in C}\sum_{f\in F} p_c \cdot d\left(c,f\right)\cdot \left(4x_c^f + 4\bar{x}_c^f\right) \\
\leq\ & 4\cdot \frac{e}{e-1} OPT_{CONF-LP-FL} < 6.33 \cdot OPT_{CONF-LP-FL}.
\end{aligned}$$

Thus we found a solution $\left(z_c^f\right)_{c\in C,f\in F}$ for the (LP-FL) whose value is at most $6.33 \cdot OPT_{CONF-LP-FL}$. However using the left-side inequality from (2) we can transform it into a solution of the (CONF-LP-FL) without losing its value. Hence we found a solution to the Universal Stochastic Facility Location which is a 6.33-approximation.

5 Multicut in the independent activation model

In the (classical version of the) Multicut problem we are given an undirected graph G with non-negative costs c_e for all $e \in E(G)$ and a set of pairs of vertices $(s_1,t_1),\ldots,(s_k,t_k)$. The goal is to erase a subset of edges F of small cost so that there is no path connecting s_c with t_c for any c. Multicut may be considered a covering problem in which the client c is covered if F contains some (s_c,t_c)-cut.

In the universal stochastic setting we are also given a probability distribution π over the subsets of $C=[1,k]$ and the solution is a mapping $\phi : C \to 2^E$ such that $\phi(c)$ forms a (s_c,t_c)-cut. The expected cost of the solution induced by a mapping ϕ equals $\mathbb{E}_{X\sim\pi}\left[\sum_{\substack{e\in\phi(c)\\ c\in X}} c_e\right]$.

We express the problem with a configuration integer program. Let $\mathcal{P}_c$ denote the family of all paths connecting s_c with t_c and let $y_B^e = 1$ mean that $B = \{c \in C : e \in \phi(c)\}$.

$$\begin{aligned}
&\min \sum_{e\in E} c_e \sum_{B\subseteq C} y_B^e \cdot g_\pi\left(B\right) && \text{(CONF-IP-MC)} && (6)\\
&\text{s.t.} \sum_{e\in P}\sum_{B\ni c} y_B^e \geq 1 && \forall_{c\in C}\,\forall_{P\in\mathcal{P}_c}. \\
&\quad y_B^e \in \{0,1\} && \forall_{e\in E}\,\forall_{B\subseteq C}.
\end{aligned}$$

We next use CONF-LP-MC to denote the linear relaxation of CONF-IP-MC. Likewise for Facility location, we will show that in the independent activation model we can reduce the universal stochastic setting to the buy-at-bulk setting, where each edge e can be either bought for price c_e to serve all the clients or be rented by client c for price $c_e \cdot p_c$. We define variables x^e to indicate that e has been bought and variables $\bar{x}_c^e$ to express the event of e being rented by c. This transition simplifies the linear program CONF-LP-MC to the following one by sacrificing an approximation factor of $\frac{e}{e-1}$.

$$\begin{aligned}
&\min \sum_{e\in E} c_e \cdot x^e + \sum_{e\in E} c_e \sum_{c\in C} p_c \cdot \bar{x}_c^e && \text{(LP-MC)} \\
&\text{s.t.} \sum_{e\in P} (x^e + \bar{x}_c^e) \geq 1 && \forall_{c\in C}\forall_{P\in\mathcal{P}_c} \\
&\quad x^e, \bar{x}_c^e \geq 0 && \forall_{e\in E}\forall_{c\in C}.
\end{aligned}$$

The proofs of the two following lemmas are similar to derivations in Sections 4.1 and 4.2 and so we omit them in this extended abstract.

▶ **Lemma 10.** *If π is an independent activation distribution, then the optimal value of LP-MP is at most $\frac{e}{e-1}$ times larger than the optimal value of CONF-LP-MP.*

▶ **Lemma 11.** *If G is a tree, then one can round a fractional solution to LP-MC to an integral one of cost at most 3 times larger. The procedure runs in polynomial time.*

In order to solve the problem on general graphs, we will embed the graph into a tree that approximately preserves the structure of cuts. The following construction has been introduced by Räcke [41]. We call a tree T decomposition tree of G if

1. the leaves of T correspond to vertices of G,
2. each edge e_t in T has weight $c^T_{e_t}$ equal to the weight of the cut it induces on $V(G)$ (we call this cut $m_T(e_t)$).

For an edge $e \in E(G)$ and a decomposition tree T we define the *relative load* of e as

$$rload_T(e) = \left(\sum_{\substack{e_t \in E(T) \\ e \in m_T(e_t)}} c^T_{e_t} \right) / c_e.$$

The main result in [41] concerns the relation between multicommodity flows in G and T_i. As our LP formulation is slightly more sophisticated we need to exploit this result in more detail. Section 2.1 in [41] describes how to find (in polynomial time) a convex combination of decomposition trees $\{\lambda_i T_i\}_{i=1}^q$ for a graph G, such that $\max_{e \in E(G)} [\sum_{i=1}^q \lambda_i rload_{T_i}(e)] = O(\log n)$.

▶ **Theorem 12.** Universal Stochastic Multicut *in the independent activation model admits a polynomial-time $O(\log n)$ approximation algorithm.*

Proof. Lemma 11 implies that a fractional solution to LP-MC over T_i can be rounded to an integer solution with ratio 3. Observe that each tree edge on a path between terminals corresponds to some cut between these terminals so any solution to Universal Stochastic Multicut on a decomposition tree induces a solution on the original graph of at most the same cost.

Let L_i denote the optimal value of LP-MC over T_i and L denote the same for G. We are going to show that $\sum_{i=1}^q \lambda_i L_i = O(L \log n)$. In particular this means that $\min_{i=1}^q L_i = O(L \log n)$. After rounding the fractional solution on T_i of the smallest value, we will obtain an integral solution for G of cost $O(L \log n)$, what entails an $O(\log n)$ approximation.

Let us consider the dual linear program of LP-MC.

$$\max \sum_{c \in C} \sum_{P \in \mathcal{P}_c} \alpha_P \qquad \text{(DP-MC)}$$

$$\text{s.t.} \sum_{c \in C} \sum_{\substack{P \in \mathcal{P}_c \\ e \in P}} \alpha_P \leq c_e \qquad \forall_{e \in E} \tag{7}$$

$$\sum_{\substack{P \in \mathcal{P}_c \\ e \in P}} \alpha_P \leq c_e \cdot p_c \qquad \forall_{c \in C} \forall_{e \in E} \tag{8}$$

$$\alpha_P \geq 0 \qquad \forall_{c \in C} \forall_{P \in \mathcal{P}_c}.$$

Feasible solutions to (DP-MC) are just multicommodity flows satisfying conditions (7)-(8). Let (β^{T_i}) be an optimal solution to (DP-MC) on a decomposition tree T_i (of value L_i) and let

(α^{T_i}) be a flow on G where a unit flow over e_t in (β^{T_i}) translates into a unit flow over $m_T(e_t)$. Note that the value of shipped commodities remains the same. Consider $(\alpha) = \sum_{i=1}^{q} \lambda_i(\alpha^{T_i})$. It is a (not necessarily feasible) solution of value $\sum_{i=1}^{q} \lambda_i L_i$.

As (β^{T_i}) routes at most $c_{e_t}^{T_i}$ flow through an edge e_t, then (α^{T_i}) routes at most $\sum_{\substack{e_t \in E(T_i) \\ e \in m_{T_i}(e_t)}} c_{e_t}^{T_i}$ flow through an edge e. Therefore constraint (7) is exceeded at most $rload_{T_i}(e)$ times in (α^{T_i}) and $\sum_{i=1}^{q} \lambda_i rload_{T_i}(e)$ times in (α). If we consider vectors $(\beta_c^{T_i})$ given by flow routed between terminals of client c, then we conclude the same for constraint (8).

After scaling (α) down times $M = \max_{e \in E(G)} [\sum_{i=1}^{q} \lambda_i rload_{T_i}(e)]$ we obtain a feasible solution to (DP-MC) for G. This means that L is no less than $\sum_{i=1}^{q} \lambda_i L_i$ divided by M. As we know that $M = O(\log n)$, the claim follows. ◀

References

1 Susanne Albers and Stefano Leonardi. On-line algorithms. *ACM Comput. Surv.*, 31(3es):4, 1999.

2 N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The Online Set Cover Problem. In *STOC'03*, pages 100–105, 2003.

3 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. A general approach to online network optimization problems. In *SODA'04*, pages 577–586, 2004.

4 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Piotr Sankowski. Online network design with outliers. *Algorithmica*, 76(1):88–109, 2016.

5 Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *STOC'03*, pages 383–388, 2003.

6 Dimitris Bertsimas and Michelangelo Grigni. Worst-case examples for the spacefilling curve heuristic for the Euclidean traveling salesman problem. *Oper. Res. Lett.*, 8(5):241–244, 1989.

7 Dimitris J. Bertsimas, Patrick Jaillet, and Amedeo R. Odoni. A priori optimization. *Oper. Res.*, 38(6):1019–1033, 1990.

8 Marcin Bienkowski, Miroslaw Korzeniowski, and Harald Räcke. A practical algorithm for constructing oblivious routing schemes. In *SPAA'03*, pages 24–33, 2003.

9 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, 1998.

10 Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing problems. In *ESA'05*, pages 689–701, 2005.

11 Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 257–269, 2005. doi:10.1007/11538462_22.

12 Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 266–278, 2009. doi:10.1007/978-3-642-02927-1_23.

13 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008. doi:10.1287/moor.1080.0330.

14 Reza Dorrigiv and Alejandro Lopez-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, 2005.

15 Esteban Feuerstein, Alberto Marchetti-Spaccamela, Frans Schalekamp, René Sitters, Suzanne van der Ster, Leen Stougie, and Anke van Zuylen. Scheduling over scenarios on two machines. In *Computing and Combinatorics – 20th International Conference, COCOON 2014, Atlanta, GA, USA, August 4-6, 2014. Proceedings*, pages 559–571, 2014. doi:10.1007/978-3-319-08783-2_48.

16 Amos Fiat and Gerhard J. Woeginger, editors. *Online algorithms*, volume 1442 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

17 Dimitris Fotakis. On the competitive ratio for online facility location. In *ICALP'03*, pages 637–652. Springer, 2003.

18 P. R. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983.

19 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *SODA'08*, pages 942–951, 2008.

20 Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 347–356, 2008. doi:10.1109/FOCS.2008.31.

21 Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. *SIAM J. Comput.*, 42(3):808–830, 2013.

22 Anupam Gupta, Mohammad Taghi Hajiaghayi, and Harald Räcke. Oblivious network design. In *SODA'06*, pages 970–979, 2006.

23 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization – 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, pages 205–216, 2013. doi:10.1007/978-3-642-36694-9_18.

24 Mohammad T. Hajiaghayi, Robert D. Kleinberg, Tom Leighton, and Harald Räcke. Oblivious routing on node-capacitated and directed graphs. In *SODA'05*, pages 782–790, 2005.

25 Mohammad Taghi Hajiaghayi, Jeong Han Kim, Tom Leighton, and Harald Räcke. Oblivious routing in directed graphs with random demands. In *STOC'05*, pages 193–201, 2005.

26 Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *EC'04*, pages 71–80, 2004.

27 Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Frank Thomson Leighton. Improved lower and upper bounds for universal tsp in planar metrics. In *SODA'06*, pages 649–658, 2006.

28 Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *SPAA'03*, pages 34–43, 2003.

29 Nicole Immorlica, David Karger, Maria Minkoff, and Vahab Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *SODA'04*, pages 684–693, 2004.

30 Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

31 Patrick Jaillet. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Oper. Res.*, 36(6):929–936, 1988.

32 Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for tsp, steiner tree, and set cover. In *STOC'05*, pages 386–395, 2005.

33 David R. Karger and Maria Minkoff. Building Steiner trees with incomplete global knowledge. In *FOCS'00*, pages 613–623, 2000.

34 Simon Korman. On the use of randomization in the online set cover problem, 2004. MSc Thesis.

35 Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. AdWords and generalized online matching. *J. ACM*, 54(5):Art. 22, 19 pp., 2007.

36 Adam Meyerson. Online facility location. In *FOCS'01*, pages 426–431. IEEE Computer Society, 2001.

37 Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Designing networks incrementally. In *FOCS'01*, pages 406–415, 2001.

38 Rolf H. Möhring, Andreas S. Schulz, and Marc Uetz. Approximation in stochastic scheduling: the power of lp-based priority policies. *J. ACM*, 46(6):924–942, 1999. doi:10.1145/331524.331530.

39 Loren K. Platzman and John J. Bartholdi, III. Spacefilling curves and the planar travelling salesman problem. *J. ACM*, 36(4):719–737, 1989.

40 Harald Räcke. Minimizing congestion in general networks. In *FOCS'02*, pages 43–52, 2002.

41 Harald Räcke. Optimal Hierarchical Decompositions for Congestion Minimization in Networks. In *STOC'08*, 2008.

42 R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *IPCO'04*, pages 101–115, 2004.

43 Frans Schalekamp and David B. Shmoys. Algorithms for the universal and a priori tsp. *Oper. Res. Lett.*, 36(1):1–3, Jan 2008.

44 David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, 2006.

45 David B. Shmoys and Kunal Talwar. A constant approximation algorithm for the a priori traveling salesman problem. In *IPCO'08*, 2008.

46 Aravind Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *SODA'07*, pages 1305–1313, 2007.

47 L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *STOC'81*, pages 263–277, 1981.

48 Martijn van Ee, Leo van Iersel, Teun Janssen, and René Sitters. A priori TSP in the scenario model. In *Approximation and Online Algorithms – 14th International Workshop, WAOA 2016, Aarhus, Denmark, August 25-26, 2016, Revised Selected Papers*, pages 183–196, 2016. doi:10.1007/978-3-319-51741-4_15.

49 Berthold Vöcking. Almost optimal permutation routing on hypercubes. In *STOC'01*, pages 530–539, 2001.

50 Jan Vondrák. *Submodularity in combinatorial optimization.* PhD thesis, Charles University, 2007.

Reusable Garbled Deterministic Finite Automata from Learning With Errors*

Shweta Agrawal[1] and Ishaan Preet Singh[2]

1 **IIT Madras, Chennai, India**
shweta@iitm.ac.in
2 **IIT Delhi, New Delhi, India**
ishaanps92@gmail.com

Abstract

We provide a single-key functional encryption scheme for Deterministic Finite Automata (DFA). The secret key of our scheme is associated with a DFA M, and a ciphertext is associated with an input $\mathbf{x}$ of *arbitrary* length. The decryptor learns $M(\mathbf{x})$ and nothing else. The ciphertext and key sizes achieved by our scheme are optimal – the size of the public parameters is independent of the size of the machine or data being encrypted, the secret key size depends only on the machine size and the ciphertext size depends only on the input size.

Our scheme achieves full functional encryption in the "private index model", namely the entire input $\mathbf{x}$ is hidden (as against $\mathbf{x}$ being public and a single bit b being hidden). Our single key FE scheme can be compiled with symmetric key encryption as in [12] to yield reusable garbled DFAs for arbitrary size inputs, that achieves machine and input privacy along with reusability under a strong simulation based definition of security.

We generalize this to a functional encryption scheme for Turing machines TMFE which has short public parameters that are independent of the size of the machine or the data being encrypted, short function keys, and input-specific decryption time. However, the ciphertext of our construction is large and depends on the worst case running time of the Turing machine (but not its description size). These provide the first FE schemes that support unbounded length inputs, allow succinct public and function keys and rely on LWE.

Our construction relies on a new and arguably natural notion of *decomposable functional encryption* which may be of independent interest.

1998 ACM Subject Classification E.3 Data Encryption

Keywords and phrases Functional Encryption, Learning With Errors, Deterministic Finite Automata, Garbled DFA

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.36

1 Introduction

Functional encryption permits *controlled disclosure* of encrypted data, enabling the evaluator to learn some authorised function of encrypted data in the clear. In functional encryption (FE), a secret key corresponds to a function f and ciphertexts correspond to strings from the domain of f. Given a function SK_f and a ciphertext $\mathsf{CT}_{\mathbf{x}}$, the decryptor learns $f(\mathbf{x})$ and nothing else. Functional encryption has found diverse applications, such as spam filtering on encrypted data [12], online dating [13], delegation of computation [16] and many others.

* A full version of the paper is available at http://www.cse.iitm.ac.in/~shwetaag/papers/dfa.pdf [1].

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 36; pp. 36:1–36:13

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

The function embedded within the secret key in FE is typically represented as a circuit. While circuits are a powerful model of computation, the circuit representation has significant drawbacks in practical scenarios. Consider the application of spam filtering on encrypted emails, where the email gateway may be given a key to test the incoming email for spam. Representing the computation as a circuit forces emails to be of a fixed length – a requirement which is ill-fitting and wasteful. Another significant drawback of the circuit model is that it incurs worst case running time on every input.

In practice, most spam filters as well as malware and intrusion detection systems are implemented using pattern matching operations represented as deterministic finite automata (DFA) [19, 14, 5, 10]. Note that in all these applications, the size of the input is highly variable and instance specific, and restricting it to be of fixed length is cumbersome. Therefore a functional encryption scheme for DFAs which supports dynamic data length would be the "right fit" in such situations. However, although functional encryption for circuits has been constructed based on the hardness of Learning With Errors (LWE) in the single key setting, it is unclear how to leverage these techniques to support the arbitrary data length required by DFAs.

1.1 Our Results

In this work, we provide a single-key functional encryption scheme for Deterministic Finite Automata (DFA). The secret key of our scheme is associated with a DFA M, and a ciphertext is associated with an input $\mathbf{x}$ of *arbitrary* length. The decryptor learns $M(\mathbf{x})$ and nothing else. The ciphertext and key sizes achieved by our scheme are optimal[1] – the public key size is independent of the machine and input size, the secret key size depends only on the machine size and the ciphertext size depends only on the input size.

Our scheme achieves full functional encryption in the "private index model", namely the entire input $\mathbf{x}$ is hidden (as against $\mathbf{x}$ being public and a single bit b being hidden). Our single key FE scheme can be compiled with symmetric key encryption as in [12] to yield reusable garbled DFAs for arbitrary size inputs, that achieves machine and input privacy along with reusability under a strong simulation based definition of security.

We generalize this to a functional encryption scheme for Turing machines TMFE which has short public parameters that are independent of the size of the machine or the data being encrypted, short function keys, and input-specific decryption time. However, the ciphertext of our construction is large and depends on the worst case running time of the Turing machine (but not its description size). These provide the first FE schemes that support unbounded length inputs, allow succinct public and function keys and rely on LWE.

Our construction relies on a new and arguably natural notion of *decomposable functional encryption* which may be of independent interest.

1.2 Related Work

Functional encryption for DFAs has received some attention already. Closest to our work is the "Attribute Based Encryption" scheme for DFAs constructed by Waters [20]. In [20], the encrypt algorithm takes as input a pair $(\mathbf{x}, b)$ where $\mathbf{x}$ may be of arbitrary size, and b is a bit. The key corresponds to a DFA machine M so that given a key for M and a ciphertext for $(\mathbf{x}, b)$, the decryptor learns the bit b if and only if M accepts $\mathbf{x}$. Note that in contrast to

[1] Up to logarithmic factors.

our work, the vector $\mathbf{x}$ is not hidden by the construction, neither is the machine M; only the bit b is hidden. On the other hand, the construction [20] can support polynomially many keys, whereas ours can only support a single key. Attrapadung [4] extended the work of Waters [20] to achieve adaptive rather than selective security. Another work that constructs Attribute Based Encryption for DFAs is by Boyen and Li [7]. However, in their construction, the input size to the DFA must be bounded in advance; avoiding this restriction is the main motivation for our work.

There are other known functional encryption systems that support unbounded size inputs, even supporting Turing machines, achieving input specific runtime and dynamic data length [11, 2, 6, 15, 8, 9]. However, the mildest assumption required by this line of work is the existence of indistinguishability obfuscation.

From standard assumptions, single key functional encryption has been constructed for all polynomial sized circuits [18, 12]. A natural approach to construct reusable garbled DFA/TM then, is to convert the machine to a circuit and leverage the constructions of [18, 12]. However, instantiating this compiler with the reusable garbled circuits construction [12] leads to a construction that cannot support dynamic data lengths, which is the main focus of this work. On the other hand, using the construction by Sahai and Seyalioglu [18] leads to a DFA/TM FE construction with large public key and ciphertext size, since the construction by [18] suffers from public key and ciphertext size that depend on the circuit size.

1.3 Our Techniques

To begin, we describe our single key FE scheme for DFA. Next, we describe how this construction may be generalized to Turing machines.

1.3.1 Single Key FE for DFA

We briefly recall how a DFA works. A DFA machine M is represented by the tuple $(Q, \Sigma, T, q_{st}, F)$ where Q is a finite set of states, Σ is a finite alphabet, $T : \Sigma \times Q \to Q$ is the transition function, q_{st} is the start state, $F \subseteq Q$ is the set of accepting states. Upon input $\mathbf{w} \in \Sigma^k$ for some arbitrary polynomial k, the machine M accepts $\mathbf{w}$ if and only if there exists a sequence of states $q_1, \ldots, q_k$ so that $q_1 = q_{st}$, $T(w_i, q_i) = q_{i+1}$ for $i \in [k-1]$, and $q_k \in F$.

To mimic the DFA computation, a natural starting point is to imagine a function key that stores the transition table of a DFA, receives as input the current (symbol, state) pair and produces as output an encryption of the next state of the computation. In more detail, say the encryptor provides encryptions of each input symbol x_i, for $i \in [|\mathbf{x}|]$, in addition to an encryption for the first (fixed) state q_{st}. Now, the function key could accept 2 inputs (x_1, q_{st}), lookup the transition table and produce an encryption of the next state q_2. Suppose this encryption can only be paired with the encryption of x_2 and none other, then we could provide (x_2, q_2) as input to the function in the next step, thus propagating the computation.

We tie together encryptions of symbol with encryptions of state via the notion of *decomposable functional encryption*. Intuitively, decomposability requires that the public key PK and the ciphertext $\mathsf{CT}_{\mathbf{y}}$ of a functional encryption scheme be decomposable into components PK_j and CT_j for $j \in [|\mathbf{y}|]$, where CT_j depends on a single deterministic bit y_j and the public key component PK_j. All components CT_j are tied together by *common randomness* used for their creation, although each CT_j may use additional independent randomness. Aside from the message dependent components, a ciphertext can contain components that are independent of the message and depend only on the common randomness. The main advantage

offered by decomposable functional encryption is that given the common randomness, each ciphertext component CT_j can be constructed independently of the rest. These components can then be joined together to create a complete ciphertext which can then be decrypted successfully. Additionally, only components that were constructed using the same randomness can be "joined", thereby preventing mix and match attacks where an adversary tries to treat mismatched symbol state pairs such (x_3, q_2) as a single legitimate input.

Now, suppose we have a decomposable functional encryption scheme for circuits. Then, we may proceed with the aforementioned strategy and divide the ciphertext into two components – the first encoding the current symbol, and the second encoding the current state. We may use the function key to generate the second component, *using the same common randomness that was used to generate the first component.*

To take this approach forward we must find a suitable decomposable functional encryption scheme for circuits – fortunately most functional encryption schemes in the literature are decomposable. In particular, we show that the the succinct single key FE by Goldwasser et al. [12] is decomposable. This scheme appears suitable for our purposes as the ciphertext and public key in this scheme are independent of circuit size.

However, note that the ciphertext of [12] suffers from *output-size dependence*, i.e. it grows linearly with the output length of the circuit. This implies that the function key may not produce an output that is proportional to the length of the ciphertext. To obtain a (single key) construction from LWE, we resolve this issue by repurposing a classic trick from Yao's garbled circuit construction, so that the output length of the circuit can be made independent of the ciphertext size, at the cost of blowing up the ciphertext size somewhat. More concretely, instead of having the circuit output a new ciphertext, the encryptor provides symmetric key encryptions of CktFE [12] ciphertext components, encrypting *all possible bit values* (nesting CktFE ciphertext inside SKE ciphertext), and the function key outputs the SKE keys to unlock the correct CktFE ciphertext components, corresponding to the bit values chosen by the key. This allows us to *select* the next state with a circuit output length independent of the ciphertext size. For more details, we refer the reader to Section 4. This provides input privacy and reusability but not machine privacy. We achieve machine privacy following ideas of [12] – please see the full version [1] for details.

1.3.2 Single key FE for Turing Machines

To extend the above construction to support Turing Machines, we must address two challenges: a) head movements should not reveal anything about the input and b) we need to write to the tape. Below we describe how to handle each challenge in turn.

To overcome the first challenge, a natural approach is to use oblivious TMs, which fix the head movement of a TM to be independent of the input. Moreover, there exist efficient transformations that convert any Turing machine M that takes time T to decide an input to an oblivious one that takes time $T \log T$ to decide the same input [17]. It remains to address the challenge of handling tape writes. Since the head movements of the TM are now fixed, the only thing that the transition function must specify is the next state, and the symbol that must be written to the current tape cell. We leverage decomposability and have the encryptor provide a ciphertext component encoding state, and another component encoding current work tape symbol *for every step in the computation.* Indeed, this forces our ciphertext to depend (linearly) on worst-case runtime of the Turing machine. All the ciphertext components for a given time step are tied together with common randomness as before. To ensure that the decryptor only learns the ciphertext components corresponding to the particular state and tape symbol that occur during computation, the encryptor encrypts

all CktFE ciphertexts with symmetric key encryption SKE. As in the case of DFA, the function key selects the appropriate SKE keys to reveal the CktFE ciphertext encoding next state and symbol to be read.

The careful reader may have noticed that the above description glosses over an important detail: the cell that is written into at step i may be next accessed at any step $j > i$, so the CktFE ciphertext at step i must encode SKE keys for some unknown future step j. Fortunately, the machinery of oblivious TMs comes to our aid again. Since in an oblivious TM, there exists a function t that computes the step that particular cell will be accessed next, in step i, in addition to selecting the state for step $i+1$ as we did in DFAs, the function key will also select the tape symbol to be read in step $t(i)$. At any step j, the appropriate SKE keys for the state were provided in step $j-1$ and for tape symbol were provided at step $i < j$ where $t(i) = j$. Hence, the decryptor at step j has the SKE keys to unlock the CktFE CT components for both state and tape symbol, which lets her proceed with the computation. For more details, please see the full version [1].

1.4 Organization of the paper

In Section 2, we define the preliminaries we require for our constructions. In Section 3, we define the notion of decomposable functional encryption. In Section 4, we provide our construction for single key FE for DFAs. In the full version [1], we provide our construction for single key functional encryption for Turing machines.

2 Definitions: FE for Deterministic Finite Automata

In this section we provide some notation and preliminaries that we require.

Functional encryption for deterministic finite automata (DFA) is defined analogously to functional encryption for circuits, except that the public parameters may not depend on the input length, which is unknown a priori. In this section, we will define single key functional encryption for DFAs.

A DFA machine M is represented by the tuple $(Q, \Sigma, T, q_{st}, F)$ where Q is a finite set of states, Σ is a finite alphabet, $T : \Sigma \times Q \to Q$ is the transition function (stored as a table), q_{st} is the start state, $F \subseteq Q$ is the set of accepting states. Upon input $\mathbf{w} \in \Sigma^k$ for some arbitrary polynomial k (not known to the setup algorithm), the machine M accepts the input if and only if there exists a sequence of states $q_1, \ldots, q_k$ so that $q_1 = q_{st}$, $T(w_i, q_i) = q_{i+1}$ for $i \in [k-1]$, and $q_k \in F$. We say $M(\mathbf{w}) = 1$ iff M accepts $\mathbf{w}$ and 0 otherwise.

2.1 Definition

Let $\mathcal{M}_\kappa : \mathcal{Q}_\kappa \times \Sigma_\kappa \to \mathcal{Q}_\kappa$ be a DFA family. A functional encryption scheme DfaFE for $\mathcal{M}$ consists of four algorithms DfaFE = (DfaFE.Setup, DfaFE.KeyGen, DfaFE.Enc, DfaFE.Dec) defined as follows.

- $\mathsf{DfaFE.Setup}(1^\kappa)$ is a p.p.t. algorithm takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{PK}, \mathsf{MSK})$.
- $\mathsf{DfaFE.KeyGen}(\mathsf{MSK}, \mathsf{M})$ is a p.p.t. algorithm that takes as input the master secret key MSK and a DFA machine M and outputs a corresponding secret key SK_M.
- $\mathsf{DfaFE.Enc}(\mathsf{PK}, \mathbf{w})$ is a p.p.t. algorithm that takes as input the master public key PK and an input message $\mathbf{w}$ and outputs a ciphertext $\mathsf{CT}_\mathbf{w}$.
- $\mathsf{DfaFE.Dec}(\mathsf{SK}_\mathsf{M}, \mathsf{CT}_\mathbf{w})$ is a deterministic algorithm that takes as input the secret key SK_M and a ciphertext $\mathsf{CT}_\mathbf{w}$ and outputs $M(\mathbf{w})$.

▶ **Definition 1** (Correctness)**.** A functional encryption scheme DfaFE is correct if for all $M \in \mathcal{M}$ and all $\mathbf{w} \in \Sigma^*$,

$$\Pr\left[\begin{array}{l}(\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{DfaFE.Setup}(1^\kappa);\\ \mathsf{DfaFE.Dec}\Big(\mathsf{DfaFE.KeyGen}(\mathsf{MSK},\mathsf{M}),\mathsf{DfaFE.Enc}(\mathsf{PK},\mathbf{w})\Big) \neq \mathsf{M}(\mathbf{w})\end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is taken over the coins of DfaFE.Setup, DfaFE.KeyGen, and DfaFE.Enc.

2.2 Security

In this section, we define simulation based security for single key FE for DFAs.

▶ **Definition 2** (FULL-SIM- Security for DFA-FE)**.** Let $\mathcal{F}_{\mathcal{M}}$ be a functional encryption scheme for a DFA family $\mathcal{M}$. For every p.p.t. adversary $A = (A_1, A_2)$ and a p.p.t. simulator Sim, consider the following two experiments:

$\mathsf{Exp}^{\mathsf{real}}_{\mathsf{DfaFE},\mathsf{A}}(1^\kappa)$:	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{DfaFE},\mathrm{Sim}}(1^\kappa)$:
1: $(\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{DfaFE.Setup}(1^\kappa)$	1: $(\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{DfaFE.Setup}(1^\kappa)$
2: $(M, st_1) \leftarrow A_1(\mathsf{PK})$	2: $(M, st_1) \leftarrow A_1(\mathsf{PK})$
3: $\mathsf{sk}_M \leftarrow \mathsf{DfaFE.KeyGen}(\mathsf{MSK},\mathsf{M})$	3: $\mathsf{sk}_M \leftarrow \mathsf{DfaFE.KeyGen}(\mathsf{MSK},\mathsf{M})$
4: $(\mathbf{x}, st) \leftarrow A_2(st_1,\mathsf{PK},\mathsf{sk}_M)$	4: $(\mathbf{x}, st) \leftarrow A_2(st_1,\mathsf{PK},\mathsf{sk}_M)$
5: $\mathsf{CT} \leftarrow \mathsf{DfaFE.Enc}(\mathsf{PK},\mathbf{x})$	5: $\tilde{\mathsf{CT}} \leftarrow \mathrm{Sim}(\mathsf{PK},\mathsf{sk}_M, M, M(\mathbf{x}), 1^{\lvert\mathbf{x}\rvert})$
6: Output (st, CT)	6: Output $(st, \tilde{\mathsf{CT}})$

The DFA functional encryption scheme $\mathcal{F}_{\mathcal{M}}$ is then said to be single query FULL-SIM secure if there exists a p.p.t. simulator Sim such that for every p.p.t. adversary $A = (A_1, A_2)$, the following two distributions are computationally indistinguishable:

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{DfaFE},\mathsf{A}}(1^\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{DfaFE},\mathrm{Sim}}(1^\kappa)\right\}_{\kappa\in\mathbb{N}}.$$

3 Decomposable Functional Encryption for Circuits

In this section, we define the notion of *decomposable functional encryption* (DFE). Decomposable functional encryption is analogous to the notion of decomposable randomized encodings [3]. Intuitively, decomposability requires that the public key PK and the ciphertext $\mathsf{CT}_{\mathbf{x}}$ of a functional encryption scheme be decomposable into components PK_i and CT_i for $i \in [|\mathbf{x}|]$, where CT_i depends on a single deterministic bit x_i and the public key component PK_i. In addition, the ciphertext may contain components that are independent of the message and depend only on the randomness.

We assume that given the security parameter, the following spaces are fixed: $\mathcal{P}$ containing public key components, $\mathcal{R}_1$, $\mathcal{R}_2$ containing randomness used for encryption and $\mathcal{C}$ containing the encoding of a single message bit. The length of the message $|\mathbf{x}|$ can be any polynomial. Formally, let $\mathbf{x} \in \{0,1\}^k$. A functional encryption scheme is said to be decomposable if there exists a deterministic function $\mathcal{E} : \mathcal{P} \times \{0,1\} \times \mathcal{R}_1 \times \mathcal{R}_2 \to \mathcal{C}$ such that:

1. The public key may be interpreted as $\mathsf{PK} = (\mathsf{PK}_1, \ldots, \mathsf{PK}_k, \mathsf{PK}_{\mathsf{indpt}})$ where $\mathsf{PK}_i \in \mathcal{P}$ for $i \in [k]$. The component $\mathsf{PK}_{\mathsf{indpt}} \in \mathcal{P}^j$ for some $j \in \mathbb{N}$.

2. The ciphertext may be interpreted as $\mathsf{CT}_{\mathbf{x}} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_k, \mathsf{CT}_{\mathsf{indpt}})$, where

$$\mathsf{CT}_i = \mathcal{E}\left(\mathsf{PK}_i, x_i, r, \hat{r}_i\right) \forall i \in [k] \quad \text{and} \quad \mathsf{CT}_{\mathsf{indpt}} = \mathcal{E}\left(\mathsf{PK}_{\mathsf{indpt}}, r, \hat{r}\right).$$

Here $r \in \mathcal{R}_1$ is common randomness used by all components of the encryption. Apart from the common randomness r, each CT_i may additionally make use of independent randomness $\hat{r}_i \in \mathcal{R}_2$.

We note that if a scheme is decomposable "bit by bit", i.e. into k components for inputs of size k, it is also decomposable into components corresponding to any partition of the interval $[k]$. Thus, we may decompose the public key and ciphertext into any $i \leq k$ components of length k_i each, such that $\sum k_i = k$. We will sometimes use $\bar{\mathcal{E}}(\mathbf{y})$ to denote the tuple of function values obtained by applying $\mathcal{E}$ to each component of a vector, i.e. $\bar{\mathcal{E}}(\mathsf{PK}, \mathbf{y}, r) \triangleq \left(\mathcal{E}(\mathsf{PK}_1, y_1, r, \hat{r}_1), \ldots, \mathcal{E}(\mathsf{PK}_k, y_k, r, \hat{r}_k)\right)$, where $|\mathbf{y}| = k$.

4 Single-Key Succinct FE for DFAs from LWE

In this section, we will construct a single key (public key) functional encryption scheme for deterministic finite automata (DFA). Our construction makes use a decomposable single key FE scheme for circuits, CktFE. In the full version [1], we show that:

▶ **Lemma 3.** *The single key, succinct functional encryption scheme for circuits by Goldwasser et al. [12], based on LWE is decomposable.*

Conceptually, we decompose the input into two components of size ℓ_1 and ℓ_2 each, where the second component is further decomposed bit by bit. We will use the first component to encrypt the current input symbol in the DFA computation and keys to select the next state in the computation, and the second component to encrypt the current state in the DFA computation. While the input symbol encoded in the first component can be treated as a unit of size ℓ_1, it will be helpful for us to represent the encoded input of size ℓ_2 bit by bit.

Thus, we have,

$$\mathsf{CktFE.PK} = (\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{PK}_{\mathsf{indpt}}) \text{ and } \mathsf{CktFE.CT} = (\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}_{\mathsf{indpt}}).$$

Now let

$$\begin{aligned}\mathsf{CktFE.Enc}(\mathsf{PK}, \mathbf{x}\|\mathbf{y}) &= (\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}_{\mathsf{indpt}}) \\ &= \left(\bar{\mathcal{E}}(\mathsf{PK}_1, \mathbf{x}, r, \hat{\mathbf{r}}_1),\ \bar{\mathcal{E}}(\mathsf{PK}_2, \mathbf{y}, r, \hat{\mathbf{r}}_2),\ \bar{\mathcal{E}}(\mathsf{PK}_{\mathsf{indpt}}, r, \hat{\mathbf{r}}_3)\right).\end{aligned}$$

We decompose

$$\bar{\mathcal{E}}(\mathsf{PK}_2, \mathbf{y}, r, \hat{\mathbf{r}}_2) = \left(\mathcal{E}(\mathsf{PK}_{2,1}, y_1, r, \hat{r}_{2,1}), \ldots, \mathcal{E}(\mathsf{PK}_{2,\ell_2}, y_{\ell_2}, r, \hat{r}_{2,\ell_2})\right).$$

Recall that $\mathcal{E} : \mathcal{P} \times \{0,1\} \times \mathcal{R}_1 \times \mathcal{R}_2 \to \mathcal{C}$ and $\bar{\mathcal{E}}(\mathbf{x})$ denotes the tuple of function values obtained by applying $\mathcal{E}$ to each coordinate. Then,

$$\text{Let} \quad |\mathbf{x}| = \ell_1, \quad |\mathbf{y}| = \ell_2, \quad \mathsf{PK}_1 \in \mathcal{P}^{\ell_1}, \quad \mathsf{PK}_2 \in \mathcal{P}^{\ell_2},$$
$$j \in \mathbb{N}, \quad \mathsf{PK}_{\mathsf{indpt}} \in \mathcal{P}^j, \quad r \in \mathcal{R}_1, \quad \hat{\mathbf{r}}_1 \in \mathcal{R}_2^{\ell_1}, \quad \hat{\mathbf{r}}_2 \in \mathcal{R}_2^{\ell_2}, \quad \hat{\mathbf{r}}_3 \in \mathcal{R}_2^{j}.$$

In what follows, we abuse notation slightly and omit mention of the independent, fresh randomness from $\mathcal{R}_2$ needed for each invocation for $\mathcal{E}$. For convenience, we club the message independent component $\mathsf{CT}_{\mathsf{indpt}}$ with CT_1 and let

$$\mathbf{c} = (\mathsf{CT}_1, \mathsf{CT}_{\mathsf{indpt}}) \text{ and } \mathbf{d} = \mathsf{CT}_2 = (\mathsf{CT}_{2,1}, \ldots, \mathsf{CT}_{2,\ell_2}).$$

Let $\mathcal{M}_\kappa : \mathcal{Q}_\kappa \times \Sigma_\kappa \to \mathcal{Q}_\kappa$ be a DFA family. For notational convenience, we will drop the subscript κ here on. Let $Q = |\mathcal{Q}|$, the size of the state space of the DFA family. Then, the single key functional encryption scheme for DFAs is constructed as follows.

DfaFE.Setup(1^κ):

Upon input the security parameter 1^κ, do:

1. Choose a symmetric key encryption scheme SKE with key space $\mathcal{K}$.
2. Define a circuit family as follows. Let $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} = (\Sigma \times \mathcal{K}^{2\log Q} \times \{0,1\}) \times \mathcal{Q}$ and $\mathcal{Y} = \mathcal{K}^{\log Q}$. We set
$$\ell_1 = \lfloor \Sigma \rfloor + \lfloor \mathcal{K}^{2\log Q} \rfloor + 1, \quad \ell_2 = \lfloor \mathcal{Q} \rfloor = \log Q$$
where $\lfloor \cdot \rfloor$ denotes size in bits. Let $\ell = \ell_1 + \ell_2$.
3. Invoke $\mathsf{CktFE.Setup}(1^\kappa, 1^\ell)$ to obtain $\mathsf{PK} = \big(\mathsf{PK}_1, (\mathsf{PK}_{2,1}, \ldots, \mathsf{PK}_{2,\log Q}), \mathsf{PK}_{\mathsf{indpt}}\big)$ and MSK.
4. Output $(\mathsf{PK}, \mathsf{MSK})$.

DfaFE.Enc($\mathsf{PK}, \mathbf{w}$):

Let $|\mathbf{w}| = k$. Note that k is arbitrary, and unknown to $\mathsf{DfaFE.Setup}$. Do the following:

1. Sample randomness $r_i \leftarrow \mathcal{R}_1$ for $i \in [k]$.
2. Sample SKE keys as follows. We define
$$\mathbf{K}_{i+1} = \Big((K^0_{(i+1,1)}, K^1_{(i+1,1)}), \ldots, (K^0_{(i+1,\log Q)}, K^1_{(i+1,\log Q)}) \Big)$$
where $K^b_{i+1,j} \leftarrow \mathcal{K}$ for $i \in [k-1]$, $j \in [\log Q]$, $b \in \{0,1\}$.
3. Define message $\mathbf{y}_i = (w_i, \mathbf{K}_{i+1}, 0)$ for $i \in [k-1]$ and $\mathbf{y}_k = (w_k, \perp, 1)$.
4. For $i \in [k]$, we define:
$$\mathbf{c}_{i,1} = \bar{\mathcal{E}}(\mathsf{PK}_1, \mathbf{y}_i, r_i), \quad \mathbf{c}_{i,2} = \bar{\mathcal{E}}(\mathsf{PK}_{\mathsf{indpt}}, r_i), \quad \mathbf{c}_i = (\mathbf{c}_{i,1}, \mathbf{c}_{i,2})\,.$$
5. Let $\mathbf{d}_1 = \bar{\mathcal{E}}\,(\mathsf{PK}_2, q_{st}, r_1)$. Here q_{st} denotes the start state of the DFA. Further, let
$$\mathbf{d}^b_{i,j} = \mathcal{E}\,(\mathsf{PK}_{2,j}, b, r_i)\ \ \forall\, i \in [2,k],\ j\ \in [\log Q],\ b \in \{0,1\}.$$
$$\mathbf{d}_{i,q} \triangleq (\mathbf{d}^{q_j}_{i,j})\ \ \forall j \in [\log Q] \text{ where } q_j \text{ is the } j\text{th bit of } q.$$
6. For $i \in [2,k]$, $j \in [\log Q]$, $b \in \{0,1\}$ encrypt each $\mathbf{d}^b_{i,j}$ using the corresponding key $K^b_{i,j}$ as:
$$\hat{\mathbf{d}}^b_{i,j} = \mathsf{SKE.Enc}(K^b_{i,j}, \mathbf{d}^b_{i,j})\,.$$
7. Choose $b_{i,j} \leftarrow \{0,1\}$ randomly for $i \in [2,k]$, $j \in [\log Q]$ and define:
$$\hat{\mathbf{D}}_{i,j} = \big(\ \hat{\mathbf{d}}^{b_{i,j}}_{i,j}, \hat{\mathbf{d}}^{\bar{b}_{i,j}}_{i,j}\ \big), \quad \hat{\mathbf{D}}_i = (\hat{\mathbf{D}}_{i,j}), \quad \hat{\mathbf{D}}_1 = \mathbf{d}_1\,.$$
8. Output $\mathsf{CT}_{\mathbf{w}} = \{\mathbf{c}_i, \hat{\mathbf{D}}_i\}$ for $i \in [k]$.

DfaFE.KeyGen(MSK, M):

Let M denote a DFA machine and T denote its transition matrix. Let T_i denote the i^{th} row of T, with format $\big((\sigma, q) \to q'\big)$ indicating that the machine enters state q' upon input symbol σ and input state q. Let $\mathsf{SK}_M = \mathsf{CktFE.Keygen}(\mathsf{MSK}, f)$ where f is defined below in Figure 1.

Function $f\big((\sigma, \mathbf{K}, \mathsf{flag}), q\big)$

1. Lookup table T for (σ, q). Say that $(\sigma, q) \to q'$. If no entry is found, output $\perp$ and exit.
2. If $\mathsf{flag} = 1$, check if q' is an accepting state. If yes, output 1, else output 0 and exit.
3. If $\mathsf{flag} = 0$, parse $\mathbf{K}$ as $\{(K_j^0, K_j^1)\}$ for $j \in [\log Q]$, $b \in \{0,1\}$. Choose the $\log Q$ keys $K_j^{q'_j}$ (for $j \in [\log Q]$), corresponding to the bits of q' and output these.

Figure 1 Function to provide keys for next state in DFA computation.

$\mathsf{DfaFE.Dec}(\mathsf{SK}_f, \mathsf{CT}_{\mathbf{w}})$:

Interpret $\mathsf{CT}_{\mathbf{w}} = (\mathbf{c}_i, \hat{\mathbf{D}}_i)_{i\in[k]}$ and let $\mathbf{d}_{1,q_1} = \hat{\mathbf{D}}_1$.

Initialize $i = 1$. While $i \le k$, do the following:

- Let $\mathsf{CT}'_i = (\mathbf{c}_i, \mathbf{d}_{i,q_i})$. Recall that $\mathbf{d}_{i,q_i} = (\mathbf{d}_{i,j}^{q_{i,j}})$ for $j \in [\log Q]$. If $i = k$, let $b \leftarrow \mathsf{CktFE.Dec}(\mathsf{SK}_f, \mathsf{CT}'_k)$. Output b and exit.
- Else let $(K_{i+1,1}, \ldots, K_{i+1,\log Q}) = \mathsf{CktFE.Dec}(\mathsf{SK}_f, \mathsf{CT}'_i)$.
- For $j \in [\log Q]$, try to decrypt each value in $\hat{\mathbf{D}}_{i+1,j}$ using obtained key $K_{i+1,j}$. Exactly one of the two ciphertexts per bit position will be decrypted, say $\hat{\mathbf{d}}_{i+1,j}^{b_j}$. Set

$$\mathbf{d}_{i+1,q_{i+1}} = \Big(\mathsf{SKE.Dec}(K_{i+1,j}, \hat{\mathbf{d}}_{i+1,j}^{b_j}) \Big) \quad \forall j \in [\log Q]\,.$$

- Increment i.

4.1 Correctness

In this section, we establish correctness of the above construction. Before we proceed with the formal argument, we provide some intuition. Note that in the encryption, the first component $\mathbf{c}_i$ encrypts message $\mathbf{y}_i$, which contains the ith input symbol, along with the set of all $2\log Q$ symmetric keys used to construct SKE encryptions of the $(i+1)^{th}$ state. In the second component, the element $\mathbf{d}_{i,j}^b$ in tuple $(\ \mathbf{d}_{i,j}^b\ )$ for $j \in [\log Q]$ and $b \in \{0,1\}$, contains an encryption of bit b, corresponding to the event that the j^{th} bit of i^{th} state is b. The set $\hat{\mathbf{D}}_i$ contains $2\log Q$ SKE encryptions of $\mathbf{d}_{i,j}^b$ under keys $K_{i,j}^b$, shuffled for each position j. Decryption at step $i-1$ provides the level i symmetric keys $K_{i,j}^{b_j}$ to unlock the $\mathbf{d}_{i,j}^{b_j}$ for the correct next state of the computation q', i.e. $b_j = q'_{i,j}$. Thus, the decryptor recovers exactly the components $\mathbf{d}_{i,j}^{b_j}$ which may be combined to create the ciphertext $\mathbf{d}_{i,q_i}$. Put together with $\mathbf{c}_i$ we get an encryption of $(w_i, \mathbf{K}_{i+1}, q_i)$ which may again be decrypted with the function key to obtain the appropriate keys to decrypt the correct $\mathbf{d}_{i+1,q_{i+1}}$.

Formally, let k denote the length of input $\mathbf{w}$ and let $q_1, \ldots, q_k$ denote the states visited by the DFA during computation. We have by correctness of decomposable functional encryption that:

$$\forall i \in [k-1],\ \mathsf{CktFE.Enc}\big(\ \mathsf{PK}, (w_i, \mathbf{K}_{i+1}, 0, q_i)\ \big) = (\mathbf{c}_i, \mathbf{d}_{i,q_i}) \quad \text{where}$$
$$\mathbf{c}_k = \Big(\bar{\mathcal{E}}\big(\mathsf{PK}_1, (w_i, \mathbf{K}_{i+1}, 0), r_i,\big), \bar{\mathcal{E}}\big(\mathsf{PK}_{\mathsf{indpt}}, r_i\big)\Big), \quad \mathbf{d}_{i,q_i} = \big(\mathcal{E}(\mathsf{PK}_{2,j}, q_{i,j}, r_i)\big)_{j\in[\log Q]}$$
$$\text{s.t. } \mathsf{CktFE.Dec}\big(\mathsf{SK}_f, (\mathbf{c}_i, \mathbf{d}_{i,q_i})\big) = \mathbf{K}_{q_{i+1}} \triangleq (K_{i+1,1}^{b_1}, \ldots, K_{i+1,\log Q}^{b_{\log Q}}) \ \text{ where } \ b_j = q_{i+1,j}.$$

Now, both elements of $\hat{\mathbf{D}}_{i+1,j}$ are attempted for decryption by $K_{i+1,j}^{b_j}$, of which only the

element encoding the correct bit $q_{i+1,j}$ is recovered. Formally, we have:

$$\hat{\mathbf{D}}_{i+1,j} = \left(\hat{\mathbf{d}}^0_{i+1,j}, \hat{\mathbf{d}}^1_{i+1,j} \right) \quad \text{and}$$
$$\mathsf{SKE.Dec}\left(K^{b_j}_{i+1,j},\ \hat{\mathbf{d}}^b_{i+1,j} \right) = \perp \ \text{ if } \ b_j \neq b, \text{ and } \mathbf{d}^{q_{i+1,j}}_{i+1,j} \quad \text{otherwise.}$$

By putting together all the components, we get by decomposability:

$$\mathbf{d}_{i+1,q_{i+1}} = \left(\mathbf{d}^{q_{i+1,j}}_{i+1,j} \right) \ \forall\, j \in [\log Q]$$

Also, since each component of $\mathbf{d}_{i+1,q_{i+1}}$ uses the same common randomness r_{i+1} as is used by $\mathbf{c}_{i+1}$, we have that $\mathsf{CT}_{i+1} = (\mathbf{c}_{i+1}, \mathbf{d}_{i+1,q_{i+1}})$, hence we may repeat while $i < k$. Finally for $i = k$,

$$\mathsf{CktFE.Enc}\big(\ \mathsf{PK}, (w_k, \perp, 1, q_k)\ \big) = (\mathbf{c}_i, \mathbf{d}_{k,q_k})$$
so that $\mathsf{CktFE.Dec}\big(\mathsf{SK}_f, (\mathbf{c}_k, \mathbf{d}_{k,q_k})\big) = 1$ iff q_k is an accepting state, 0 otherwise.

Efficiency. We note that the public key size of our scheme is the public key size of CktFE [12] with message length $\ell = O(\log Q + \log|\Sigma| + \log Q \cdot \log|\mathcal{K}|$ which is polynomial in the security parameter κ. The ciphertext size is $O(|\mathbf{w}| \cdot \log Q)$ and the secret key size is $O(|M|)$ (ignoring polynomials in the security parameter).

4.2 Proof of Security

We proceed to show that our construction is secure. Formally:

▶ **Theorem 4.** *Assume that the underlying* CktFE *scheme satisfies* FULL-SIM *security according to definition (please see* [1]*). Then the construction for* DfaFE *achieves* FULL-SIM *security as defined in Definition 2.*

Proof. We proceed to construct a simulator DfaFE.Sim as required by Definition 2. The simulator receives $(\mathsf{PK}, \mathsf{SK}_M, M, M(\mathbf{w}), 1^{|\mathbf{w}|})$ and does the following:

1. Assign the bit $b = M(\mathbf{w})$, and construct the circuit f corresponding to M as defined in Figure 1 in the description of DfaFE.KeyGen.
2. Let $\mathsf{CktFE.SK_f} = \mathsf{SK_M}$ and invoke $\mathsf{CktFE.Sim}\big(\ \mathsf{PK}, f, \mathsf{CktFE.SK}_f, b\ \big)$ to receive $\tilde{\mathsf{CT}}_k$ where we may express $\tilde{\mathsf{CT}}_k = (\tilde{\mathbf{c}}_k, \tilde{\mathbf{d}}_{k,q_k})$ and $\tilde{\mathbf{d}}_{k,q_k} = (\tilde{\mathbf{d}}_{k,j})$ for $j \in [\log Q]$.
3. For $(i = k, i \geq 1, i--)$, do:
   a. If $i = 1$, set $\mathsf{Sim}.\hat{\mathbf{D}}_1 = \tilde{\mathbf{d}}_{1,q_1}$ and exit.
   b. Sample key $\mathbf{K}^*_i = (K^*_{i,1}, \ldots, K^*_{i,\log Q}) \leftarrow \mathcal{K}^{\log Q}$ and let
   $$\mathsf{Sim}.\hat{\mathbf{d}}_{i,j} = \mathsf{SKE.Enc}(K^*_{i,j}, \tilde{\mathbf{d}}_{i,j}) \ \ \forall j \in [\log Q]\,.$$
   c. Sample $\tilde{b}_{i,j} \leftarrow \{0,1\}$ and assign $\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_{i,j}}_{i,j} = \mathsf{Sim}.\hat{\mathbf{d}}_{i,j}$.
   d. Choose another $\log Q$ keys $\tilde{K}_{i,1}, \ldots \tilde{K}_{i,\log Q} \leftarrow \mathcal{K}^{\log Q}$ and compute
   $$\mathsf{Sim}.\hat{\mathbf{d}}^{\bar{b}_{ij}}_{i,j} = \mathsf{SKE.Enc}(\tilde{K}_{i,j}, 0^{|\tilde{\mathbf{d}}_{i,j}|}) \ \ \forall\, j \in [\log Q]\,.$$
   e. Let $\mathsf{Sim}.\hat{\mathbf{D}}_{i,j} = \big(\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_{i,j}}_{i,j},\ \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{b}_{i,j}}_{i,j}\big)$ and $\mathsf{Sim}.\hat{\mathbf{D}}_i = \big(\ \mathsf{Sim}.\hat{\mathbf{D}}_{i,j}\ \big)$ for $j \in [\log Q]$.
   f. Let $(\tilde{\mathbf{c}}_{i-1}, \tilde{\mathbf{d}}_{i-1,q_{i-1}}) = \mathsf{CktFE.Sim}(\mathsf{PK}, f, \mathsf{SK}_f, \mathbf{K}^*_i)$.
4. Output the ciphertext as $\mathsf{CT_w} = (\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \ldots, \tilde{\mathbf{c}}_k, \mathsf{Sim}.\hat{\mathbf{D}}_1, \ldots, \mathsf{Sim}.\hat{\mathbf{D}}_k)$.

4.2.1 Analysis of Simulator

Correctness of the simulator DfaFE.Sim can be easily established using correctness of the simulator CktFE.Sim and the semantic security of SKE. Let us say that the DFA M visits states $q_1, \ldots, q_k$ while computing on input $\mathbf{w}$ where $|\mathbf{w}| = k$.

1. We have by correctness of CktFE.Sim that:

$$\left\{ \mathsf{CT}_k \leftarrow \mathsf{CktFE.Enc}\big(\mathsf{PK}, (w_k, \bot, 1, q_k)\big) \stackrel{c}{\approx} \tilde{\mathsf{CT}}_k \leftarrow \mathsf{CktFE.Sim}\big(\mathsf{PK}, f_M, \mathsf{SK}_f, b\big) \right\}.$$

   By decomposability, $\mathsf{CT}_k = (\mathbf{c}_k, \mathbf{d}_{k,q_k})$ where $\mathbf{d}_{k,q_k} = (\mathbf{d}_{k,j}^{b_j})$ for $j \in [\log Q]$ and $b_j = q_{k,j}$ defined as the j^{th} bit of state q_k. Similarly, $\tilde{\mathsf{CT}}_k = (\tilde{\mathbf{c}}_k, \tilde{\mathbf{d}}_{k,q_k})$ where $\tilde{\mathbf{d}}_{k,q_k}$ may be decomposed as $(\tilde{\mathbf{d}}_{k,j})$ for $j \in [\log Q]$. Let $i = k$.

2. We now establish that $(\hat{\mathbf{d}}_{i,j}^{b_j} \stackrel{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}_{i,j})$ where $b_j = q_{i,j}$ and $j \in [\log Q]$.
   - **a.** We have that in algorithm DfaFE.Enc,

$$\mathbf{K}_i = \Big((K^0_{(i,1)}, K^1_{(i,1)}), \ldots, (K^0_{(i,\log Q)}, K^1_{(i,\log Q)}) \Big)$$

     where $K^b_{i,j} \leftarrow \mathcal{K}$ for $j \in [\log Q]$. We also have, for $j \in [\log Q]$, $b \in \{0,1\}$:

$$\hat{\mathbf{d}}^b_{i,j} = \mathsf{SKE.Enc}(K^b_{i,j}, \mathbf{d}^b_{i,j}) \tag{4.1}$$

   - **b.** In simulator DfaFE.Sim:

$$\mathbf{K}^*_i = (K^*_{i,1}, \ldots, K^*_{i,\log Q}) \leftarrow \mathcal{K}^{\log Q} \quad \text{and}$$

$$\mathsf{Sim}.\hat{\mathbf{d}}_{i,j} = \mathsf{SKE.Enc}(K^*_{i,j}, \tilde{\mathbf{d}}_{i,j}) \;\; \forall j \in [\log Q]\,.$$

     Hence, since $\mathbf{d}^{b_j}_{i,j} \stackrel{c}{\approx} \tilde{\mathbf{d}}_{i,j}$ and the symmetric keys are picked using the same distribution in each case, we have that $(\hat{\mathbf{d}}^{b_j}_{i,j} \stackrel{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}_{i,j})$ where $b_j = q_{i,j}$ and $j \in [\log Q]$.

3. We now establish that $(\hat{\mathbf{d}}^{\bar{b}_j}_{i,j} \stackrel{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{b}_j}_{i,j})$ where $j \in [\log Q]$.
   - **a.** Construction of $\hat{\mathbf{d}}^{\bar{b}_j}_{i,j}$ is described in Equation 4.1.
   - **b.** For the latter, DfaFE.Sim samples $\tilde{b}_j$ and sets $\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_j}_{i,j} = \mathsf{Sim}.\hat{\mathbf{d}}_{i,j}$. Next, it samples another $\log Q$ keys $\tilde{K}_{i,1}, \ldots \tilde{K}_{i,\log Q} \leftarrow \mathcal{K}^{\log Q}$ and computes

$$\mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_j}_{i,j} = \mathsf{SKE.Enc}(\tilde{K}_{i,j}, 0^{|\tilde{\mathbf{d}}_{i,j}|}) \;\; \forall\, j \in [\log Q]\,.$$

     By semantic security of SKE, we have that $(\hat{\mathbf{d}}^{\bar{b}_j}_{i,j} \stackrel{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{b}_j}_{i,j})$.

4. Next, we show that $\hat{\mathbf{D}}_i \stackrel{c}{\approx} \mathsf{Sim}.\hat{\mathbf{D}}_i$. For $i = 1$, we have by definitions of $\hat{\mathbf{D}}_1$ and $\mathsf{Sim}.\hat{\mathbf{D}}_1$, that the above holds. For $i > 1$, in DfaFE.Enc, we have $b_{i,j} \leftarrow \{0,1\}$ and

$$\hat{\mathbf{D}}_{i,j} = \big(\hat{\mathbf{d}}^{b_{i,j}}_{i,j}, \hat{\mathbf{d}}^{\bar{b}_{i,j}}_{i,j} \big)\,.$$

   In DfaFE.Sim, we have $\tilde{b}_{i,j} \leftarrow \{0,1\}$ and

$$\mathsf{Sim}.\hat{\mathbf{D}}_{i,j} = \big(\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_{i,j}}_{i,j},\ \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_{i,j}}_{i,j}\big)\,.$$

   Since $\hat{\mathbf{D}}_i = (\hat{\mathbf{D}}_{i,j})$ and $\mathsf{Sim}.\hat{\mathbf{D}}_i = \big(\ \mathsf{Sim}.\hat{\mathbf{D}}_{i,j}\ \big)$ for $j \in [\log Q]$, we have that $\hat{\mathbf{D}}_i \stackrel{c}{\approx} \mathsf{Sim}.\hat{\mathbf{D}}_i$.

5. Let $i = i - 1$. Now, we have by correctness of $\mathsf{CktFE.Sim}$,

$$\left\{\mathsf{CT}_i \leftarrow \mathsf{CktFE.Enc}\big(\mathsf{PK}, (w_i, \mathbf{K}_{i+1}, 0, q_i)\big) \overset{c}{\approx} \tilde{\mathsf{CT}}_i \leftarrow \mathsf{CktFE.Sim}\big(\mathsf{PK}, f_M, \mathsf{SK}_f, \mathbf{K}^*_{i+1}\big)\right\}.$$

   By decomposability, $\mathsf{CT}_i = (\mathbf{c}_i, \mathbf{d}_{i,q_i})$ where $\mathbf{d}_{i,q_i} = (\mathbf{d}_{i,j}^{q_{i,j}})$ for $j \in [\log Q]$. Also, $\tilde{\mathsf{CT}}_i = (\tilde{\mathbf{c}}_i, \tilde{\mathbf{d}}_{i,q_i})$ where $\tilde{\mathbf{d}}_{i,q_i} = (\tilde{\mathbf{d}}_{i,j})$ for $j \in [\log Q]$. If $i > 1$, go to step 2. For $i = 1$, we have by definitions of $\hat{\mathbf{D}}_1$ and $\mathsf{Sim}.\hat{\mathbf{D}}_1$, that $(\mathbf{c}_1, \hat{\mathbf{D}}_1) \overset{c}{\approx} (\tilde{\mathbf{c}}_1, \mathsf{Sim}.\hat{\mathbf{D}}_1)$.
6. Now, a straightforward hybrid argument yield that:

$$\left\{(\mathbf{c}_1, \hat{\mathbf{D}}_1), (\mathbf{c}_2, \hat{\mathbf{D}}_2), \ldots, (\mathbf{c}_k, \hat{\mathbf{D}}_k)\right\} \overset{c}{\approx} \left\{(\tilde{\mathbf{c}}_1, \mathsf{Sim}.\hat{\mathbf{D}}_1), (\tilde{\mathbf{c}}_2, \mathsf{Sim}.\hat{\mathbf{D}}_2), \ldots, (\tilde{\mathbf{c}}_k, \mathsf{Sim}.\hat{\mathbf{D}}_k)\right\}$$

   as desired.

◀

Reusable Garbled DFA. In the full version [1] we show how to compile the above construction with symmetric key encryption to obtain the first construction of reusable garbled DFAs from standard assumptions.

5 Single Key Functional Encryption for Turing Machines

In the full version [1], we provide the construction of single key functional encryption for Turing machines. Our construction has short public parameters that are independent of the size of the machine or the data being encrypted, short function keys, and input-specific decryption time. However, the ciphertext of our construction is large and depends on the worst case running time of the Turing machine (but not its description size).

While the large ciphertext size of our TMFE construction restricts its utility for practical applications, we emphasize that the parameters obtained by our TMFE construction are not implied by previous work to the best of our knowledge (please see the full version [1] for a detailed discussion about previous work). To improve the ciphertext size of our construction, while allowing succinct keys, dynamic data length and input specific run time is an interesting open problem.

References

1 Shweta Agrawal and Ishaan Preet Singh. Reusable garbled deterministic finite automata from learning with errors, full version. `http://www.cse.iitm.ac.in/~shwetaag/papers/dfa.pdf`, 2017.

2 Prabhanjan Ananth and Amit Sahai. Functional encryption for turing machines. In *Theory of Cryptography*, pages 125–153. Springer, 2016.

3 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. *SIAM J. Comput.*, 43(2):905–929, 2014.

4 Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, 2014.

5 Domagoj Babić, Daniel Reynaud, and Dawn Song. Malware analysis with tree automata inference. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, 2011.

6 Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In *STOC*, 2015.

7 Xavier Boyen and Qinyi Li. Attribute-Based Encryption for Finite Automata from LWE. In *Provable Security*, pages 247–267. Springer, 2015.

8 Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, 2015.

9 Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Computation-trace indistinguishability obfuscation and its applications. *IACR Cryptology ePrint Archive*, 2015, 2015.

10 Sanjeev Das, Hao Xiao, Yang Liu, and Wei Zhang. Online malware defense using attack behavior model. In *IEEE International Symposium on Circuits and Systems, ISCAS 2016, Montréal, QC, Canada, May 22-25, 2016*, pages 1322–1325, 2016.

11 Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

12 Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

13 Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate Encryption for Circuits from LWE. In *Crypto*, 2015.

14 Christopher L. Hayes and Yan Luo. Dpico: A high speed deep packet inspection engine using compact finite automata. In *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, ANCS'07, pages 195–203, 2007.

15 Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, 2015.

16 Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, 2012.

17 Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM (JACM)*, 26(2):361–381, 1979.

18 Amit Sahai and Hakan Seyalioglu. Worry-free encryption: Functional encryption with public keys. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS'10, 2010.

19 Daniele Paolo Scarpazza, Oreste Villa, and Fabrizio Petrini. Peak-performance dfa-based string matching on the cell processor. In *21th International Parallel and Distributed Processing Symposium (IPDPS), Proceedings, 26-30 March 2007, Long Beach, California, USA*, pages 1–8, 2007.

20 Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012.

Round-Preserving Parallel Composition of Probabilistic-Termination Cryptographic Protocols*

Ran Cohen[†1], Sandro Coretti[‡2], Juan Garay[3], and Vassilis Zikas[4]

1 **School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
cohenran@tauex.tau.ac.il
2 **Courant Institute of Mathematical Sciences, New York University, New York, NY, USA**
corettis@nyu.edu
3 **Yahoo Research, Sunnyvale, CA, USA**
garay@yahoo-inc.com
4 **Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA**
vzikas@cs.rpi.edu

Abstract

An important benchmark for secure multi-party computation (MPC) protocols is their *round complexity*. For several important MPC tasks, (tight) lower bounds on the round complexity are known. However, for some of these tasks, such as broadcast, the lower bounds can be circumvented when the termination round of every party is not *a priori* known, and simultaneous termination is not guaranteed. Protocols with this property are called *probabilistic-termination (PT)* protocols.

Running PT protocols in parallel affects the round complexity of the resulting protocol in somewhat unexpected ways. For instance, an execution of m protocols with constant expected round complexity might take $O(\log m)$ rounds to complete. In a seminal work, Ben-Or and El-Yaniv (Distributed Computing '03) developed a technique for parallel execution of arbitrarily many broadcast protocols, while preserving expected round complexity. More recently, Cohen *et al.* (CRYPTO '16) devised a framework for universal composition of PT protocols, and provided the first composable parallel-broadcast protocol with a simulation-based proof. These constructions crucially rely on the fact that broadcast is "privacy free," and do not generalize to arbitrary protocols in a straightforward way. This raises the question of whether it is possible to execute *arbitrary* PT protocols in parallel, without increasing the round complexity.

In this paper we tackle this question and provide both feasibility and infeasibility results. We construct a round-preserving protocol compiler, secure against a minority of actively corrupted parties, that compiles arbitrary protocols into a protocol realizing their parallel composition, while having a black-box access to the underlying *protocols*. Furthermore, we prove that the same cannot be achieved, using known techniques, given only black-box access to the *functionalities* realized by the protocols, unless merely security against semi-honest corruptions is required, for which case we provide a protocol.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Cryptographic protocols, secure multi-party computation, broadcast

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.37

* The full version of this paper can be found at the *IACR Cryptology ePrint Archive* [15], http://eprint.iacr.org/2017/364.

† Research supported by ERC starting grant 638121.

‡ Author supported by NSF grants 1314568 and 1319051.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 37; pp. 37:1–37:15

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

1 Introduction

Secure multi-party computation (MPC) [52, 30] allows a set of parties to jointly perform a computation on their inputs, in such a way that no coalition of cheating parties can learn any information beyond what is revealed by their outputs (privacy) or affect the outputs of the computation in any way other than by choosing their own inputs (correctness). Since the first seminal works on MPC [52, 30, 6, 12, 50], it has been studied in a variety of different settings and for numerous security notions: there exist protocols secure against passively corrupted (aka semi-honest) parties and against actively corrupted (aka malicious) parties; the underlying network can be synchronous or asynchronous; and the required security guarantees can be information-theoretic or computational – -to name but a few of the axes along which the MPC task can be evaluated.

The prevalent model for the design of MPC protocols is the synchronous model, where the protocol proceeds in rounds. In this setting, the round complexity, i.e., the number of rounds it takes for a protocol to deliver outputs, is arguably the most important efficiency metric. Tight lower bounds are known on the round complexity of several MPC tasks. For example, for the well-known problems of Byzantine agreement (BA) and broadcast [48, 44], it is known that any protocol against an active attacker corrupting a linear fraction of the parties has linear round complexity [25, 23]. This result has quite far-reaching consequences as, starting with the seminal MPC works mentioned above, a common assumption in the design of secure protocols has been that the parties have access to a broadcast channel, which they potentially invoke in every round. In reality, such a broadcast channel might not be available and would have to be implemented by a broadcast protocol designed for a point-to-point network. It follows that even though the round complexity of many MPC protocols is linear in the multiplicative depth of the circuit being computed, their actual running time depends on the number of parties, when executed over point-to-point channels.

The above lower bound on the number rounds for BA holds when all honest parties are required to complete the protocol together, at the same round [22]. Indeed, *randomized* BA protocols that circumvent this lower bound and run in *expected constant* number of rounds (cf. [4, 49, 24, 26, 41, 45]) do not provide simultaneous termination, i.e., once a party completes the protocol's execution it cannot know whether all honest parties have also terminated or if some honest parties are still running the protocol; in particular, the termination round of each party is not a priori known. A protocol with this property is said to have *probabilistic termination (PT)*.

As pointed out by Ben-Or and El-Yaniv [5], when several such PT protocols are executed in parallel, the expected round complexity of the combined execution might no longer be constant (specifically, might not be equal to the maximum of the expected running times of the individual protocols). Indeed, when m protocols, whose termination round is geometrically distributed (and so, have constant expected round complexity), are run in parallel, the expected number of rounds that elapse before all of them terminate is $\Theta(\log m)$ [14]. While an elegant mechanism was proposed in [5] for implementing parallel calls to broadcast such that the total expected number of rounds remains constant, it did not provide any guarantees to remain secure under composition, raising questions about its usability in a higher-level protocol (such as the MPC setting described above). Such a shortcoming was recently addressed by Cohen *et al.* [14] who provided a framework for universal composition of PT protocols (building upon the universal-composition framework of [8]). An application of their result was the first composable protocol for parallel broadcast (with a simulation-based proof) that can be used for securely replacing broadcast channels in arbitrary protocols, and whose round complexity is constant in expectation.

Indeed, an immediate application of the composable parallel-broadcast protocol from [14] is plugging it into broadcast-model MPC protocols in order to obtain point-to-point protocols with a round complexity that is independent of the number of parties. In the information-theoretic setting, this approach yields protocols whose round complexity depends on the depth of the circuit computing the function [6, 12, 50, 18], whereas in the computational setting, assuming standard cryptographic assumptions, this approach yields expected-constant-round protocols [43, 3, 20, 40, 1, 29, 31, 46]. However, the resulting point-to-point protocols have probabilistic-termination on their own. The techniques used for composing PT broadcast protocols in parallel crucially rely on the fact that broadcast is a privacy-free functionality, and a naïve generalization of this approach to arbitrary PT protocols fails to be secure. This raises the question of whether it is possible to execute *arbitrary* PT protocols in parallel, without increasing the round complexity.

We remark that circumventing lower bounds on round complexity is just one of the areas where such PT protocols have been successfully used. Indeed, randomizing the termination round has been proven to be a very useful technique in circumventing impossibilities and improving efficiency for many cryptographic protocols. Notable examples include *non-committing encryption* [21], cryptographic protocols designed for rational parties [34, 27, 47, 2, 32, 28], concurrent zero-knowledge protocols [9, 13] and parallel repetition of interactive arguments [33, 35]. The rich literature on such protocols motivates a thorough investigation of their security and composability. As mentioned above, in [14] the initial foundations were laid out for such an investigation, but what was proven for arbitrary PT protocols was a round-preserving *sequential* composition theorem, leaving parallel composition as an open question.

Our contributions. In this work, we investigate the issue of parallel composition for *arbitrary* protocols with probabilistic termination. In particular, we develop a compiler such that given functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$ and protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$, where for every $i \in [\mathtt{M}]$, protocol π_i realizes $\mathcal{F}_i$ (possibly using correlated randomness as setup[1]), then the compiled protocol realizes the parallel composition of the functionalities, denoted $(\mathcal{F}_1 \| \cdots \| \mathcal{F}_{\mathtt{M}})$.

Our compiler uses the underlying protocols in a black-box manner,[2] is robust (i.e., secure without abort) and resilient against a computationally unbounded active adversary, adaptively corrupting up to $t < n/2$ parties (which is optimal [50]). Moreover, our compiler is round-preserving, meaning that if the maximal (expected) round complexity of each protocol is μ, then the expected round complexity of the compiled protocol is $O(\mu)$. For example, if each protocol π_i has constant expected round complexity, then so does the compiled protocol. Recall that this task is quite complicated even for the simple case of BA (cf. [5, 14]). For arbitrary functionalities it is even more involved, since as we show, the approach from [5] cannot be applied in a functionally black-box way in this case. Thus, effectively, our result is the first round-preserving parallel composition result for arbitrary multi-party protocols/tasks with probabilistic termination.

We now describe the ideas underlying our compiler. In [5] (see also [14]), a round-preserving parallel-broadcast protocol was constructed by iteratively running, for a constant number of rounds, *multiple* instances of BA protocols (each instance is executed multiple

[1] A trusted setup phase is needed for implementing broadcast in the honest-majority setting. As shown in [17, 16] some interesting functions can be computed without such a setup phase.

[2] Following [37], by a black-box access to a protocol we mean a black-box usage of a semi-honest MPC protocol computing its next-message function.

times in parallel, in a batch), hoping that at least one execution of every BA instance will complete. By choosing the multiplicity suitably, this would occur with constant probability, and the process need therefore be repeated a constant expected number of times only.

At first sight it might seem that this idea can be applied to arbitrary tasks, but this is not the case. Intuitively, the reason is that if the tasks that we want to compose in parallel have privacy requirements, then making the parties run them in (parallel) "batches" with the same input might compromise privacy, since the adversary will be able to use different inputs and learn multiple outputs of the function(s). This issue is not relevant for broadcast, because it is a "privacy-free" functionality; the adversary may learn the result of multiple computations using the same inputs for honest parties, without compromising security.

To cope with the above issue, our parallel-composition compiler generalizes the approach of [5] in a privacy-preserving manner. At a high level, it wraps the batching technique by an MPC protocol which restricts the parties to use the same input in all protocols for the same function. In particular, the compiler is defined in the *Setup-Commit-then-Prove* hybrid model [10, 39], which allows each party to commit to its input values and later execute multiple instances of every protocol, each time proving that the same input value is used in all executions.

The constructions in [10, 39] for realizing the Setup-Commit-then-Prove functionality are designed for the dishonest-majority setting and therefore allow for a premature abort. Since we assume an honest majority, we require security without abort. A possible way around would be, as is common in the MPC literature, to restart the protocol upon discovering some cheating or add for each abort a recovery round; this, however, would induce a linear overhead (in the number of parties) on the round complexity of the protocol.

Instead, in order to recover from a misbehavior by corrupted parties, we modify the Setup-Commit-then-Prove functionality and secret-share every committed random string between all the parties, using an error-correcting secret-sharing scheme (aka robust secret sharing [50, 19, 11]). In case a party is identified as cheating, every party broadcasts the share of the committed randomness corresponding to that party, reconstructs the correlated randomness for that party, and locally computes the messages corresponding to this party in every instance of every protocol. We also prove that the modified Setup-Commit-then-Prove functionality can be realized in a constant number of rounds, thus yielding no (asymptotic) overhead on the round complexity of the compiler.

Next, given that using only black-box access to the protocols π_i, it is possible to compile them into a protocol that implements the parallel composition $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ of the functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$ realized by protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$, we investigate the question of whether there exists a protocol that securely realizes $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ given only black-box access to the *functionalities* $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$,[3] but not to protocols realizing them. This question is only sensible if asked for an entire *class* of functionalities (cf. [51]), since otherwise a protocol may always ignore the functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$ and implement $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ from scratch.

On the one hand, we prove that against semi-honest corruptions, there indeed exists a protocol for parallel composition of arbitrary functionalities $\mathcal{F}_i$ in a functionally black-box manner. On the other hand, in the case of active corruptions, we devise a class of functionalities for which, when using a generalization of the "batching" technique from [5],

[3] Loosely speaking, a functionally black-box protocol, as defined in [51], is a protocol that can compute a function f without knowing the code of f, i.e., given an oracle access to the function f. Note that in this model, each ideal functionality $\mathcal{F}_i$ has an oracle access to the function f_i it computes.

such a black-box transformation is not possible in the presence of a single active corrupted party. More precisely, (1) a naïve call to each of the ideal functionalities $\mathcal{F}_i$ until termination will not be round-preserving, (2) it is impossible to compute the parallel composition without calling every ideal functionality, and (3) using the same input value in more than one call to any of the ideal functionalities will break privacy.[4] This negative result validates our choice of a *protocol* compiler, and is evidence that such a task, if at all possible, would require entirely new techniques.

We phrase our results using the framework for composition of protocols with probabilistic termination [14], extending it (a side result of independent interest) to include parallel composition, reactive functionalities (to capture the Setup-Commit-then-Prove functionality), and the higher corruption threshold of $t < n/2$.

Organization of the paper. The rest of the paper is organized as follows. In Section 2 we describe the network model, the basics of the probabilistic-termination framework by Cohen *et al.* [14], and other tools that are used throughout the paper. We start Section 3 with the extensions to the PT framework, followed by the protocol that achieves round-preserving parallel composition for arbitrary functionalities in a functionally black-box manner against semi-honest adversaries. Section 4 is dedicated to active corruptions; first, the round-preserving protocol-black-box construction, followed by the negative result on round-preserving functionally black-box composition in the case of active corruptions. Due to space limitations, finer details of the model and PT framework, our extension of the PT framework to the honest-majority setting, and proofs, will only appear in the full version of the paper.

2 Model and Preliminaries

2.1 Synchronous Protocols in UC

We consider synchronous protocols in the model of Katz *et al.* [42], which is designed on top of the universal composability framework of Canetti [8]. More specifically, we consider n parties $P_1, \ldots, P_n$ and a computationally unbounded, adaptive *t-adversary* that can dynamically corrupt up to t parties during the protocol execution. Synchronous protocols in [42] are protocols that run in a hybrid model where parties have access to a simple "clock" functionality. This functionality keeps an indicator bit, which is switched once *all honest parties* request the functionality to do so, i.e., once all honest parties have completed their operations for the current round. In addition, all communication is done over bounded-delay secure channels, where each party requests the channel to fetch messages that are sent to him, such that the adversary is allowed to delay the message delivery by a bounded and a priori known number of fetch requests. Stated differently, once the sender has sent some message, it is guaranteed that the message will be delivered within a known number of activations of the receiver. For simplicity, we assume that every message is delivered within a single fetch request. A more detailed overview of [42] can be found in the full version.

[4] We note that our negative result does not contradict Ishai et al. [38, 36] who constructed two-round protocols with guaranteed output delivery for $n \geq 4$ and $t = 1$ without broadcast. Indeed, the protocols in [38, 36] are non-black-box with respect to the function to be computed.

2.2 The Probabilistic-Termination Framework

Cohen *et al.* [14] extended the UC framework to capture protocols with probabilistic termination, i.e., protocols without a fixed output round and without simultaneous termination. This section outlines their techniques; additional details can be found in the full version.

Canonical synchronous functionalities. The main idea behind modeling probabilistic termination is to separate the functionality to be computed from the round complexity that is required for the computation. The atomic building block in [14] is a functionality template called a *canonical synchronous functionality (CSF)*, which is a simple two-round functionality with explicit (one-round) input and (one-round) output phases. The functionality $\mathcal{F}_{\text{CSF}}$ has two parameters: (1) a (possibly) randomized function f that receives $n+1$ inputs (n inputs from the parties and one additional input from the adversary) and (2) a leakage function l that determines what information about the input values is leaked to the adversary.

$\mathcal{F}_{\text{CSF}}$ proceeds in two rounds: in the first (input) round, all the parties hand $\mathcal{F}_{\text{CSF}}$ their input values, and in the second (output) round, each party receives its output. Whenever some input is submitted to $\mathcal{F}_{\text{CSF}}$, the adversary is handed some leakage function of this input; the adversary can use this leakage when deciding the inputs of corrupted parties. Additionally, he is allowed to input an extra message, which – depending on the function f – might affect the output(s).

Wrappers and traces. Computation with probabilistic termination is captured by defining *output-round randomizing wrappers*. Such wrappers address the issue that while an ideal functionality abstractly describes a protocol's task, it does not describe its round complexity. Each wrapper is parametrized by a distribution (more precisely, an efficient probabilistic sampling algorithm) D that may depend on a specific protocol implementing the functionality. The wrapper samples a round $\rho_{\mathsf{term}} \leftarrow D$, by which all parties are guaranteed to receive their outputs. Two wrappers are considered: the first, denoted $\mathcal{W}_{\text{strict}}$, ensures in a strict manner that all (honest) parties terminate together in round ρ_{term}; the second, denoted $\mathcal{W}_{\text{flex}}$, is more flexible and allows the adversary to deliver outputs to individual parties at any time before round ρ_{term}.

As pointed out in [14], it is not sufficient to inform the simulator $\mathcal{S}$ about the round ρ_{term}. In many cases, the wrapper should explain to $\mathcal{S}$ *how* this round was sampled; concretely, the wrapper provides $\mathcal{S}$ with the random coins that are used to sample ρ_{term}. In particular, $\mathcal{S}$ learns the entire *trace* of calls to ideal functionalities that are made by the protocol in order to complete by round ρ_{term}. A trace basically records which hybrids were called by a protocol's execution, and in a recursive way, for each hybrid, which hybrids would have been called by a protocol realizing that hybrid.The recursion ends when the base case is reached, i.e., when the protocol is defined using the *atomic* functionalities that are "assumed" by the model.[5] Formally, a trace is defined as follows:

▶ **Definition 1** (Traces). A *trace* is a rooted tree of depth at least 1, in which all nodes are labeled by functionalities and where every node's children are *ordered*. The root and all internal nodes are labeled by wrapped CSFs (by either of the two wrappers), and the leaves are labeled by unwrapped CSFs. The *trace complexity* of a trace T, denoted $c_{\mathsf{tr}}(T)$, is the number of leaves in T. Moreover, denote by $\mathsf{flex}_{\mathsf{tr}}(T)$ the number nodes labeled by flexibly wrapped CSFs in T.

[5] The atomic functionalities considered in this work are the CSFs for the point-to-point communication functionality $\mathcal{F}_{\text{SMT}}$ and the correlated-randomness functionality for broadcast $\mathcal{F}_{\text{CORR-BC}}$.

Sequential composition of probabilistic-termination protocols. When a set of parties execute a probabilistic-termination protocol, or equivalently, invoke a flexibly wrapped CSF, they might get out-of-sync and start the next protocol in different rounds. The approach in [14] for dealing with sequential composition is to start by designing simpler protocols, that are in a so-called *synchronous normal form*, where the parties remain in-sync throughout the execution, and next, compile these protocols into slack-tolerant protocols.

▶ **Definition 2** (Synchronous normal form)**.** Let $\mathcal{F}_1, \ldots, \mathcal{F}_m$ be CSFs. A synchronous protocol π in the $(\mathcal{F}_1, \ldots, \mathcal{F}_m)$-hybrid model is in *synchronous normal form (SNF)* if in every round exactly one ideal functionality $\mathcal{F}_i$ is invoked by all honest parties, and in addition, no honest party hands inputs to other CSFs before this instance halts.

SNF protocols are designed as an intermediate step only, since the hybrid functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_m$ are two-round CSFs, and, in general, cannot be realized by real-world protocols. In order to obtain protocols that can be realized in the real world, [14] introduced *slack-tolerant* variants of both the strict and the flexible wrappers, denoted $\mathcal{W}_{\text{sl-strict}}$ and $\mathcal{W}_{\text{sl-flex}}$. These wrappers are parametrized by a slack parameter $c \geq 0$ and can be used even if parties provide inputs within $c+1$ consecutive rounds (i.e., they tolerate input slack of c rounds); furthermore, the wrappers ensure that all honest parties obtain output within two consecutive rounds (i.e., they reduce the slack to $c = 1$). Next, [14] constructed compilers to convert any SNF protocol realizing a wrapped CSF $\mathcal{W}^{D}_{\text{strict}}(\mathcal{F})$ (resp., $\mathcal{W}^{D}_{\text{flex}}(\mathcal{F})$) into a (non SNF) protocol realizing $\mathcal{W}^{D',c}_{\text{sl-strict}}(\mathcal{F})$ (resp., $\mathcal{W}^{D',c}_{\text{sl-flex}}(\mathcal{F})$), using wrapped CSFs as hybrids. The compilers maintain the security and the asymptotic (expected) round complexity of the original SNF protocols. At the same time, the compilers take care of any potential slack that is introduced by the protocol and ensure that the resulting protocol can be safely executed even if the parties do not start the protocol simultaneously.

Finally, in [14], the authors also provided protocols for realizing wrapped variants of the atomic CSF functionality for secure point-to-point communication. This suggested the following design paradigm for realizing a wrapped functionality $\mathcal{W}_{\text{sl-strict}}(\mathcal{F})$ (resp., $\mathcal{W}_{\text{sl-flex}}(\mathcal{F})$): First, construct an SNF protocol for realizing $\mathcal{W}_{\text{strict}}(\mathcal{F})$ (resp., $\mathcal{W}_{\text{flex}}(\mathcal{F})$) using CSF hybrids $\mathcal{F}_1, \ldots, \mathcal{F}_m$. Next, for each of the non-atomic hybrids $\mathcal{F}_i$, show how to realize $\mathcal{W}_{\text{strict}}(\mathcal{F}_i)$ (resp., $\mathcal{W}_{\text{flex}}(\mathcal{F}_i)$) using CSF hybrids $\mathcal{F}'_1, \ldots, \mathcal{F}'_{m'}$. Proceed in this manner until all CSF hybrids are atomic functionalities. Finally, repeated applications of the composition theorems above yield a protocol for $\mathcal{W}_{\text{sl-strict}}(\mathcal{F})$ (resp., $\mathcal{W}_{\text{sl-flex}}(\mathcal{F})$) using only atomic functionalities as hybrids.

2.3 A Lemma on Termination Probabilities

The following lemma, which will be used in our positive results, provides a constant lower bound on the probability that when running simultaneously (i.e., in parallel) $\mathtt{N}$ copies of $\mathtt{M}$ probabilistic-termination protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$, at least one copy of each π_i will complete after $\mathtt{R}$ rounds, for suitable choices of $\mathtt{N}$ and $\mathtt{R}$. The proof of the lemma appears in the full version.

▶ **Lemma 3.** *Let $M, N, R \in \mathbb{N}$. For $i \in [M]$ and $j \in [N]$, let X_{ij} be independent random variables over the natural numbers, such that $X_{i1}, \ldots, X_{iN}$ are identically distributed with expectation μ_i, for every $i \in [M]$. Denote $Y_i = \min\{X_{i1}, \ldots, X_{iN}\}$ and $\mu = \max\{\mu_1, \ldots, \mu_M\}$.*

Then, for any constant $0 < \epsilon < 1$, if $R > \mu$ and $N > \frac{\log(M/\epsilon)}{\log(R/\mu)}$, then $\Pr[\forall i : Y_i < R] \geq 1 - \epsilon$.

3 Round-Preserving Parallel Composition: Passive Security

In this section, we show that round-preserving parallel composition is feasible, in a functionally black-box manner, facing semi-honest adversaries. To that end we first extend, in Section 3.1, the probabilistic-termination framework [14] to capture the notions of functionally black-box protocols [51] and of parallel composition of canonical synchronous functionalities. The passively secure protocol is presented in Section 3.2.

3.1 Functionally Black-Box Protocols and Parallel Composition

Functionally black-box protocols. We formalize the notion of functionally black-box protocols of Rosulek [51] in the language of canonical synchronous functionalities. As in [51], we focus on secure function evaluation. The SFE functionality $\mathcal{F}_{\text{SFE}}^{g}$, parametrized by an n-party function g, is defined as the CSF $\mathcal{F}_{\text{CSF}}^{f_{\text{SFE}},l_{\text{SFE}}}$, where $f_{\text{SFE}}(x_1,\ldots,x_n,a) = g(x_1,\ldots,x_n)$ (i.e., computes the function g while ignoring the adversary's input a) and the leakage function is $l_{\text{SFE}}(x_1,\ldots,x_n) = (|x_1|,\ldots,|x_n|)$. The following definition explains what we mean by a protocol that realizes the secure function evaluation functionality in a black-box way with respect to the function g.

▶ **Definition 4.** Let $\mathcal{C} = \{g\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n\}$ be a class of n-party functions. Denote by $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ the CSF, implemented as an (uninstantiated) oracle machine that in order to compute $f_{\text{SFE}}^{\mathcal{C}}(x_1,\ldots,x_n,a)$, queries the oracle with $(x_1,\ldots,x_n)$ and stores the response $(y_1,\ldots,y_n)$. The leakage function $l_{\text{SFE}}(x_1,\ldots,x_n) = (|x_1|,\ldots,|x_n|)$ is unchanged.

Then, a protocol $\pi = (\pi_1,\ldots,\pi_n)$ is a functionally black-box (FBB) protocol for (a wrapped version of) $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$, if for every $f \in \mathcal{C}$, the protocol $\pi^f = (\pi_1^f,\ldots,\pi_n^f)$ UC-realizes $\mathcal{F}_{\text{SFE}}^{f}$.

Parallel Composition of CSFs. The parallel composition of CSFs is defined in a natural way as the CSF that evaluates the corresponding functions in parallel.

▶ **Definition 5.** Let $f_1,\ldots,f_{\mathtt{M}}$ be n-input functions. We define the $(n\cdot\mathtt{M})$-input function $(f_1 \parallel \cdots \parallel f_{\mathtt{M}})$ as follows. Upon input $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$, where each $\boldsymbol{x}_i$ is an $\mathtt{M}$-tuple $(x_i^1,\ldots,x_i^{\mathtt{M}})$, the output is the $\mathtt{M}$-tuple defined as

$$(f_1 \parallel \cdots \parallel f_{\mathtt{M}})(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n) = \big((y_1^1,\ldots,y_1^{\mathtt{M}}),\ldots,(y_n^1,\ldots,y_n^{\mathtt{M}})\big)\,,$$

where $(y_1^j,\ldots,y_n^j) = f_j(x_1^j,\ldots,x_n^j)$.

Let $\mathcal{F}_{\text{CSF}}^{f_1,l_1},\ldots,\mathcal{F}_{\text{CSF}}^{f_{\mathtt{M}},l_{\mathtt{M}}}$ be CSFs, denote $\mathcal{F}_i = \mathcal{F}_{\text{CSF}}^{f_i,l_i}$. The parallel composition of $\mathcal{F}_1,\ldots,\mathcal{F}_{\mathtt{M}}$, denoted as $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$, is the CSF defined by the function $(f_1 \parallel \cdots \parallel f_{\mathtt{M}})$ and the leakage function $(l_1 \parallel \cdots \parallel l_{\mathtt{M}})$.

3.2 Passively Secure FBB Parallel-Composition Protocol

The underlying idea of our protocol is based on a simplified form of the parallel-broadcast protocol of Ben-Or and El-Yaniv [5]. The protocol proceeds in iterations, where in each iteration, the parties invoke, in parallel and using the same input values, sufficiently many instances of each (oracle-aided) ideal functionality, but only for a constant number of rounds. If some party received an output in at least one invocation of every ideal functionality, it distributes all output values and the protocol completes; otherwise, the protocol resumes with another iteration. This protocol retains privacy for deterministic functions,[6] since the

[6] Although the result holds for deterministic functionalities, we note that using standard techniques every functionality can be transformed to an equivalent deterministic functionality in a black-box way.

adversary is semi-honest, and so corrupted parties will provide the same input values to all instances of each ideal functionality.

Intuitively, during the simulation of the protocol, the simulator should imitate every call for every ideal functionality towards the adversary. A subtle issue is that in order to do so, the simulator must know the exact trace that is sampled by each instance of each ideal functionality during the execution of the real protocol. Therefore, it is indeed essential for the simulator to receive the *random coins* used to sample the trace for the entire protocol, by the ideal functionality computing the parallel composition (cf. Section 2.2). By defining the trace-distribution sampler in a way that consists of all (potential) sub-traces for every instance of every ideal functionality, the simulator can induce the exact random coins used to sample the correct sub-trace for every ideal functionality that is invoked.

▶ **Theorem 6.** *Let $\mathcal{C}_1, \ldots, \mathcal{C}_M$ be (deterministic-)function classes, let $\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}, \ldots, \mathcal{F}_{\text{SFE}}^{\mathcal{C}_M}$ be oracle-aided secure function evaluation functionalities, and let $t < n/2$. Let $D_1, \ldots, D_M$ be distributions, such that for every $j \in [M]$, the round complexity of $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_j})$ has expectation μ_j. Denote $\mu = \max\{\mu_1, \ldots, \mu_M\}$.*

Then, $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \| \cdots \| \mathcal{F}_{\text{SFE}}^{\mathcal{C}_M})$, for some distribution D with expectation $\mu' = O(\mu)$, can be UC-realized by an FBB protocol in the $(\mathcal{F}_{\text{SMT}}, \mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}), \ldots, \mathcal{W}_{\text{flex}}^{D_M}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_M}))$-hybrid model, with information-theoretic security, in the presence of an adaptive, semi-honest t-adversary, assuming that all honest parties receive their inputs at the same round.

In particular, if for every $j \in [M]$, the expectation μ_j is constant, then μ' is constant.

The proof of Theorem 6 can be found in the full version.

4 Round-Preserving Parallel Composition: Active Security

In this section, we consider security against active adversaries. First, in Section 4.1, we show how to compute the parallel composition of probabilistic-termination functionalities, in a round-preserving manner, using a black-box access to *protocols* realizing the individual functionalities. In Section 4.2, we investigate the question of whether there exists a *functionally* black-box round-preserving malicious protocol for the parallel composition of probabilistic functionalities, and show that for a natural extension of protocols, following the techniques from [5], this is not the case – i.e., there exist functions such that no such protocol with black-box access to them can compute their parallel composition, in a round-preserving manner, tolerating even a single adversarial party.

4.1 Feasibility of Round-Preserving Parallel Composition

In this section, we show how to compile multiple protocols, realizing probabilistic-termination functionalities, into a single protocol that realizes the parallel composition of the functionalities, in a round-preserving manner, and while only using black-box access to the underlying protocols. We start by providing a high-level description of the compiler.

The compiler receives as input protocols $\pi_1, \ldots, \pi_M$, where each protocol π_j is defined in the point-to-point model, in which the parties are given correlated randomness in a secure setup phase, i.e., in the $(\mathcal{F}_{\text{SMT}}, \mathcal{F}_{\text{CORR}}^{D_j^{\text{corr}}})$-hybrid model.[7] It follows that the next-message function for each party in each protocol is a deterministic function that receives the input

[7] This captures, for example, broadcast-model protocols, where the broadcast channel is realized using an expected-constant-round protocol.

value, correlated randomness, private randomness and history of incoming messages, and outputs a vector of n messages to be sent in the following round (one message for each party). In particular, we note that the entire transcript of the protocol is fixed once the input value, correlated randomness and private randomness of each party are determined.

The underlying ideas of the compiler are inspired by the constructions in [5, 14], where a round-preserving parallel-broadcast protocol was constructed by iteratively running, for a constant number of rounds, multiple instances of BA protocols (each instance is executed multiple times in parallel), until at least one execution of every BA instance is completed. This approach is suitable for computing "privacy-free" functionalities, where the adversary may learn the results of multiple computations using the same inputs for honest parties, without compromising security. However, when considering the parallel composition of arbitrary functions, running two instances of a protocol *using the same inputs* will violate privacy, since the adversary can use different inputs to learn multiple outputs of the function.

The parallel-composition compiler generalizes the above approach in a privacy-preserving manner. The compiler follows the GMW paradigm [30] and is defined in the Setup-Commit-then-Prove hybrid model [10, 39], which generates committed correlated randomness for the parties and ensures that all parties follow the protocol specification. This mechanism allows each party to commit to its input values and later execute multiple instances of each protocol, while proving that the same input value is used in all executions. For simplicity and without loss of generality, we assume that each function is deterministic and has a public output. In this case it is ensured that if two parties receive output values in two executions of π_j, then they receive the same output value. The private random coins that are used in each execution only affect the termination round, but not the output value. Using this simplification, we can remove the leader-election phase from the output-agreement technique in [5, 14] and directly use the termination technique from Bracha [7].

Another obstacle is to recover from corruptions without increasing the round complexity. Indeed, in case some party misbehaves, e.g., by using different input values in different instances of the same protocol π_j, then the Setup-Commit-then-Prove functionality ensures that all honest parties will identify the cheating party. In this case, the parties cannot recover by, for example, backtracking and simulating the cheating party, as this will yield a round complexity that is linear in the number of parties. Furthermore, the protocol must resume in a way such that all instances of a specific protocol π_j will use the same input value that the identified corrupted party used throughout the protocol's execution until it misbehaved (since the cheating party might have learned an output value in one of the executed protocols).

To this end, we slightly adjust the Setup-Commit-then-Prove functionality and secret-share every committed random string r_i (the correlated randomness for party P_i) among all the parties, using an error-correcting secret-sharing scheme. Note that this can be done information theoretically as we assume an honest majority [50, 19, 11]. In case a cheating party is identified, every party broadcasts the share of the committed randomness corresponding to that party, reconstructs this party's correlated randomness and from that point onwards, locally computes the messages corresponding to this party in every instance of every protocol. Using this approach, every round in the original protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$ is expanded by a constant number of rounds, and the overall round complexity is preserved.

▶ **Theorem 7.** *Let $\mathcal{F}_1, \ldots, \mathcal{F}_M$ be CSFs, let $t < n/2$, and let $c \geq 1$. Let $\pi_1, \ldots, \pi_M$ be SNF protocols such that for every $j \in [M]$, protocol π_j UC-realizes $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ with expected round complexity μ_j and information-theoretic security, in the $(\mathcal{F}_{\text{SMT}}, \mathcal{F}_{\text{CORR}}^{D_j^{\mathsf{corr}}})$-hybrid model (for a distribution D_j and a distribution D_j^{corr} in NC^0), in the presence of an adaptive, malicious t-adversary, assuming that all honest parties receive their inputs at the same round. Denote $\mu = \max\{\mu_1, \ldots, \mu_M\}$.*

Then, $\mathcal{W}^{D,c}_{\text{sl-flex}}(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_M)$*, for some distribution* D *with expectation* $\mu' = O(\mu)$*, can be UC-realized with information-theoretic security by a protocol* π *in the* $(\mathcal{F}_{\text{SMT}}, \mathcal{F}_{\text{CORR-BC}})$*-hybrid model, in the same adversarial setting, assuming that all honest parties receive their inputs within* $c+1$ *consecutive rounds. In addition, protocol* π *requires only black-box access to the protocols* π_j*.*

In particular, if for every $j \in [M]$*,* μ_j *is constant, then* D *has constant expectation.*

The proof of Theorem 7 can be found in the full version.

4.2 An Impossibility of FBB Round-Preserving Parallel Composition

In this section, we prove that for a natural class of protocols, following and/or extending in various ways the techniques from Ben-Or and El-Yaniv [5],[8] there exist functions such that no protocol can compute their parallel composition in a round-preserving manner, while accessing the functions in a black-box way, tolerating even a single adversarial party. Although this is not a general impossibility result, it indicates that the batching approach of [5] is limited to semi-honest security (cf. Section 3) and/or functionally white-box transformations.

We observe that this impossibility serves as an additional justification for the optimality of our protocol-black-box parallel composition (cf. Section 4.1). Indeed, on the one hand, it formally confirms the generic observation that the natural parallel composition of a set of PT functionalities does not preserve their round complexity. On the other hand, and most importantly, it proves that all existing techniques for composing PT functionalities in parallel in the natural (FBB) manner fail in preserving the round complexity. Hence, the only known existing round-preserving composition for such functionalities are the protocol-black-box compiler presented in Section 4.1 or more inefficient non-black-box techniques. The wideness of the class of excluded protocols by our impossibility result justifies our conjecture that there exists no round-preserving FBB protocol for parallel composition of PT functionalities. Proving this conjecture is in our opinion a very interesting research direction.

We first argue informally why the approach of [5], cannot be directly extended to privacy-sensitive functions. The idea in [5] for allowing each of the n parties to broadcast its value is to have each of the n parties participate in $m = O(\log n)$ parallel invocations (hereafter called *batches*, to avoid confusion with the goal of parallel broadcast for different messages) of broadcast as sender with the same input. Each of those batches is executed in parallel for a fixed (constant) number of rounds (for the same broadcast message); this increases the probability that sufficiently many parties receive output from each batch. At the end of each batch execution, the parties check whether they jointly hold the output, and if not, they repeat the computation of the batches. It might seem that this idea can be applied to arbitrary tasks, but this is not the case. The reason is that this idea fails if the functionality has any privacy requirements, is that the adversary can input different values on different calls of the functionality within a batch and learn more information on the input.

Batched parallel composition. The above issue with privacy appears whenever a function is invoked twice in the same round on the same inputs from honest parties. Indeed, in this case the adversary can use different inputs to each invocation and learn information as sketched above. The same attack can be extended to composition-protocols which invoke the function in two different rounds ρ and ρ'; as long as the adversary knows these rounds

[8] To our knowledge, the only known techniques for round-preserving parallel composition are those of Ben-Or and El-Yaniv [5] and are only for the specific case of Byzantine agreement.

he can still launch the above attack on privacy. Generalizing the claim even further, for specific classes of functions, it suffices that there are two (possibly different) functions which are evaluated on the same inputs in rounds ρ and ρ'. This excludes protocols that might attempt to avoid using some functionality $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ by invoking some other $\mathcal{W}_{\text{flex}}^{D_{j'}}(\mathcal{F}_{j'})$.

To capture the above generalization, we define the class of *batched-parallel composition* protocols: A protocol π implementing the PT parallel composition $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ in the $(\mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_1), \ldots, \mathcal{W}_{\text{flex}}^{D_{\mathtt{M}}}(\mathcal{F}_{\mathtt{M}}))$-hybrid model (for some distributions $D, D_1, \ldots, D_{\mathtt{M}}$) is a *batched-parallel composition* protocol if it has the following structure: It proceeds in rounds, where in each round the protocol might initiate (possibly multiple) calls to any number of the hybrid functionalities $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ and/or continue calls that were initiated in previous rounds. Furthermore, there exist two publicly known protocol rounds ρ and ρ', and indices $j, j', \ell \in [\mathtt{M}]$, such that for the input vector $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ that π gives to $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ (where $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^{\mathtt{M}})$) the following properties are satisfied:

1. In round ρ the functionality $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ is called on input $\boldsymbol{x}^\ell = (x_1^\ell, \ldots, x_n^\ell)$ and at least two of its rounds are executed.
2. In round ρ' the functionality $\mathcal{W}_{\text{flex}}^{D_{j'}}(\mathcal{F}_{j'})$ is also called on input $\boldsymbol{x}^\ell$ and at least two of its rounds are executed.

We next show that the there are classes of functions $\mathcal{C}_1, \ldots, \mathcal{C}_{\mathtt{M}}$ such and for any protocol π that securely computes the parallel composition $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \parallel \cdots \parallel \mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}})$ while given hybrid access to PT functionalities $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$ the following properties hold *simultaneously*:

1. π has to call each of the hybrids $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$ (for at least 2 rounds each).[9]
2. The naïve solution of π calling each of the $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$'s in parallel until they terminate is not round-preserving (for an appropriate choice of D_i's.)
3. π cannot be a batched-parallel composition protocol.

The above shows that the classes $\mathcal{C}_1, \ldots, \mathcal{C}_{\mathtt{M}}$ not only exclude the existence of a batched-parallel composition protocol, but they also exclude all other known solutions. This implies that for this classes of functions, every known approach – and generalizations thereof – fail to compute the parallel composition of the corresponding functionality in an FBB and round-preserving manner. In the full version we prove the following theorem.

▶ **Theorem 8.** *Let $\mathtt{M} = O(\kappa)$. There exist n-party function classes $\mathcal{C}_1, \ldots, \mathcal{C}_{\mathtt{M}}$ and distributions $D_1, \ldots, D_{\mathtt{M}}$, such that the following properties hold in the presence of a malicious adversary corrupting any one of the parties:*

1. *The protocol that calls each $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$ in parallel (once) until termination is not round-preserving (its round complexity is asymptotically higher than of the distributions D_i).*
2. *Any $(\mathcal{F}_{\text{SMT}}, \mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}), \ldots, \mathcal{W}_{\text{flex}}^{D_{\mathtt{M}}}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}}))$-hybrid protocol for computing $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \parallel \cdots \parallel \mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}})$ has to make a meaningful call (i.e., a call that executes at least two rounds) to each PT hybrid $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$.*
3. *There exists no functionally black-box batched-parallel composition protocol for computing $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \parallel \cdots \parallel \mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}})$ in the $(\mathcal{F}_{\text{SMT}}, \mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}), \ldots, \mathcal{W}_{\text{flex}}^{D_{\mathtt{M}}}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}}))$-hybrid model, where D has (asymptotically) the same expectation as $D_1, \ldots, D_{\mathtt{M}}$.*

References

1 Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and

[9] Note that this does not mean that π is not round preserving as the calls might be in parallel.

interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, April 2012.

2 Gilad Asharov and Yehuda Lindell. Utility dependence in correct and fair rational secret sharing. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 559–576. Springer, August 2009.

3 Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

4 Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *2nd ACM PODC*, pages 27–30. ACM Press, August 1983.

5 Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

6 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

7 Gabriel Bracha. An asynchronou [(n-1)/3]-resilient consensus protocol. In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *3rd ACM PODC*, pages 154–162. ACM Press, August 1984.

8 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

9 Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires omega (log n) rounds. In *33rd ACM STOC*, pages 570–579. ACM Press, July 2001.

10 Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

11 Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 195–208. Springer, April 2012.

12 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

13 Kai-Min Chung, Rafael Pass, and Wei-Lung Dustin Tseng. The knowledge tightness of parallel zero-knowledge. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 512–529. Springer, March 2012.

14 Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269. Springer, August 2016.

15 Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. Cryptology ePrint Archive, Report 2017/364, 2017. URL: `http://eprint.iacr.org/2017/364`.

16 Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. Characterization of secure multiparty computation without broadcast. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 596–616. Springer, January 2016.

17 Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 466–485. Springer, December 2014.

18 Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 311–326. Springer, May 1999.

19 Ronald Cramer, Ivan Damgård, and Serge Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 503–523. Springer, August 2001.

20 Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 378–394. Springer, August 2005.

21 Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, August 2000.

22 Danny Dolev, Rüdiger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM*, 37(4):720–741, 1990.

23 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

24 Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

25 Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

26 Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 211–220. ACM Press, July 2003.

27 Georg Fuchsbauer, Jonathan Katz, and David Naccache. Efficient rational secret sharing in standard communication networks. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 419–436. Springer, February 2010.

28 Juan A. Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *54th FOCS*, pages 648–657. IEEE Computer Society Press, October 2013.

29 Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, February 2014.

30 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

31 S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, August 2015.

32 Adam Groce and Jonathan Katz. Fair computation with rational players. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 81–98. Springer, April 2012.

33 Iftach Haitner. A parallel repetition theorem for any interactive argument. In *50th FOCS*, pages 241–250. IEEE Computer Society Press, October 2009.

34 Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: Extended abstract. In László Babai, editor, *36th ACM STOC*, pages 623–632. ACM Press, June 2004.

35 Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. An efficient parallel repetition theorem. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 1–18. Springer, February 2010.

36 Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 359–378. Springer, August 2015.

37 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

38 Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594. Springer, August 2010.

39 Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 369–386. Springer, August 2014.

40 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, August 2008.

41 Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, August 2006.

42 Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498. Springer, March 2013.

43 Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

44 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

45 Silvio Micali. Fast and furious byzantine agreement. In *ITCS 2017*, January 2017.

46 Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 735–763. Springer, May 2016.

47 Shien Jin Ong, David C. Parkes, Alon Rosen, and Salil P. Vadhan. Fairness with an honest minority and a rational majority. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 36–53. Springer, March 2009.

48 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

49 Michael O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409, November 1983.

50 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989.

51 Mike Rosulek. Must you know the code of f to securely compute f? In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 87–104. Springer, August 2012.

52 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

Cryptanalysis of Indistinguishability Obfuscations of Circuits over GGH13*†

Daniel Apon[1], Nico Döttling[‡2], Sanjam Garg[2], and Pratyay Mukherjee[4]

1 **University of Maryland, College Park, MD, USA**
dapon@cs.umd.edu
2 **University of California, Berkeley, CA, USA**
nicodoettling@berkeley.edu
3 **University of California, Berkeley, CA, USA**
sanjamg@berkeley.edu
4 **Visa Research, Palo Alto, CA, USA**
pratyay85@gmail.com

Abstract

Annihilation attacks, introduced in the work of Miles, Sahai, and Zhandry (CRYPTO 2016), are a class of polynomial-time attacks against several candidate indistinguishability obfuscation ($i\mathcal{O}$) schemes, built from Garg, Gentry, and Halevi (EUROCRYPT 2013) multilinear maps. In this work, we provide a *general* efficiently-testable property for two single-input branching programs, called *partial inequivalence,* which we show is sufficient for our variant of annihilation attacks on several obfuscation constructions based on GGH13 multilinear maps.

We give examples of pairs of natural NC^1 circuits, which – when processed via Barrington's Theorem – yield pairs of branching programs that are partially inequivalent. As a consequence we are also able to show examples of "bootstrapping circuits," (albeit somewhat artificially crafted) used to obtain obfuscations for all circuits (given an obfuscator for NC^1 circuits), in certain settings also yield partially inequivalent branching programs. Prior to our work, no attacks on any obfuscation constructions for these settings were known.

1998 ACM Subject Classification E.3 Data Encryption

Keywords and phrases Obfuscation, Multilinear Maps, Cryptanalysis.

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.38

1 Introduction

An obfuscator is a program compiler which hides all partial implementation details of a program, intuitively. This is formalized via the notion of indistinguishability obfuscation [9]: we say an obfuscator $\mathcal{O}$ is an indistinguishability obfuscator if it holds for every pair C_0, C_1 of functionally equivalent circuits (i.e. computing the same function) that $\mathcal{O}(C_0)$ and $\mathcal{O}(C_1)$

* This is the extended abstract of the full version [4] which can be found at https://eprint.iacr.org/2016/1003. Most proofs are deferred to the full version.

† Research supported in part from 2017 AFOSR YIP Award, DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, NSF CRII Award 1464397, and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

‡ Nico Döttling was supported by a postdoc fellowship of the German Academic Exchange Service (DAAD).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 38; pp. 38:1–38:16

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

are indistinguishable. A recent surge of results has highlighted the importance of this notion: virtually "any cryptographic task" can be achieved assuming indistinguishability obfuscation and one-way functions [34].

All known candidate constructions of indistinguishability obfuscation, e.g. [25, 8, 6], are based on multilinear-maps [24, 21, 27][1], which have been the subjects of various attacks [16, 18, 15, 29, 19]. Among them, the attacks (e.g. [24, 29]) on GGH13 [24] multilinear maps required explicit access to "low-level" encodings of zero, or differently represented low-level encodings of zero, e.g. [18]; such low-level zero-encodings do not appear naturally in obfuscation constructions. Recently Miles, Sahai, and Zhandry [32] introduced a new class of polynomial-time[2] attacks without requiring low-level zeros against several obfuscation constructions [12, 8, 3, 31, 33] and [7], when instantiated with the GGH13 multilinear maps.

More specifically, Miles et al. [32] exhibit two simple *branching programs* (and also programs padded with those) that are functionally equivalent, yet their BGKPS-obfuscations (put forward by Barak et al. in [8]) and similar constructions [12, 3, 31, 33, 7] are efficiently distinguishable.[3] However, the branching programs considered there, in particular the *all-identity* branching program, do not appear "in the wild". More specifically, obfuscation constructions for circuits first convert an NC^1 circuit into a branching program (e.g. via Barrington's transformation) that possibly results in programs with complex structures, even if one starts with simple circuits. This brings us to the following open question:

Is it possible to attack obfuscations of complex branching programs generated from NC^1 *circuits?*

1.1 Our Contributions

In this work, we are able to answer the above question affirmatively. In particular, our main contributions are:

- We first define a *general* and efficiently-testable property of two single-input[4] branching programs called *partial inequivalence* (discussed below) and demonstrate an annihilation attack against BGKPS-like obfuscations of any two (large enough) branching programs that satisfy this property.
- Next, using implementation in Sage [35] (see the full version for details on the implementation) we give explicit examples of pairs of (functionally equivalent) natural NC^1 circuits, which when processed via Barrington's Theorem yield pairs of branching programs that are partially inequivalent – and thus, attackable.
- As a consequence of the above result, we are also able to show that the "bootstrapping circuit(s)" technique used to boost $i\mathcal{O}$ for NC^1 to $i\mathcal{O}$ for $\mathsf{P/poly}$, for a certain choice of the universal circuit (albeit artificially crafted), yield partially inequivalent branching programs in a similar manner – and are, thus, also attackable.

---

[1] The work of [2] might be seen as an exception to this: Assuming the (non-explicit) existence of indistinguishability obfuscation, they provide an explicit construction of an indistinguishability obfuscator.

[2] Several subexponential-time or quantum-polynomial-time [22, 1, 17] attacks on GGH13 multilinear maps also have been considered. We do not consider these in this paper.

[3] To avoid repetitions, from now on we will refer to the obfuscation constructions of [8, 12, 3, 31, 33] by BGKPS-like constructions that use single-input branching programs.

[4] The branching programs, where any pair of matrices in the sequence depends on a single input location, are called single-input branching programs. Such branching programs naturally evolve from Barrington's transformation on circuits.

	Branching Programs	NC^1 Circuits (Barrington's)	NC^1-to-P/poly [25, 5] [11, 28]
GGHRSW[25]	⊗	○	○
BGKPS-like constructions [12, 8, 3] [33, 31, 7]	×	⊗	⊗
Obfuscations from weak multilinear maps [26, 23]	○	○	○

Figure 1 The Attack Landscape for GGH13-based Obfuscators. In all cases, the multilinear map is [24]. ○ means no attack is known. × means a prior attack is known, and we present more general attacks for this setting. ⊗ means we give the first known attack in this setting and ⊗ means a new attack is discovered concurrently to ours (namely [13]).

Our general partial inequivalence condition is broad and seems to capture a wide range of natural single-input branching programs. However, we require the program to be large enough.[5] Additionally, we need the program to output 0 on a large number of its inputs.

Finally, our new annihilation attacks are essentially based on linear system solvers and thus quite *systematic*. This is in contrast with the attacks of Miles et al. [32] which required an exhaustive search operation rendering it hard to extend their analysis for branching programs with natural structural complexity. Therefore, at a conceptual level, our work enhances the understanding of the powers and the (potential) limits of annihilation attacks.

One limitation of our technique is that they do *not* extend to so-called dual-input branching programs. We leave it as an interesting open question.

A Concurrent and Independent work

Concurrent and independent to our work,[6] Chen et al. [13] provides a polynomial time attack against the GGHRSW construction [25] based on GGH13 (and also GGH15 [27]) maps that works for so-called "input-partitioning" branching programs. Nonetheless, their attacks are not known to extend [14] for complex branching programs evolved from NC^1 circuits (e.g. via Barrington's Transformation). Hence, our work stands as the *only* work that breaks obfuscations of NC^1 circuits based on GGH13 till date.

Change in Obfuscation landscape

Given our work and the work of Chen et al. [13] the new *attack landscape against GGH13-based obfuscators* is depicted in Figure 1. We refer the reader to [2, Figure 13] for the state of the art on obfuscation constructions based on CLT13 and GGH15 multilinear maps.

[5] Note that, for our implementation we consider circuits that are quite small, only depth 3, and the resulting Barrington programs are of length 64. However, using the implementation we then "boost" the attack to a much larger NC^1 circuits that suffice for the real-world attack (discussed in the full version) to go through.

[6] The first draft of our full version [4] appeared online concurrently as their first draft [13]. At the same time another independent work [20] appeared that provided attacks against several CLT13 based obfuscators for a broader class of programs.

1.2 Technical Overview

Below, after providing some additional backgrounds on multilinear maps and known attacks, we provide an overview of our annihilation attacks.

Multilinear Maps: Abstractly

As a first approximation, one can say that a cryptographic multilinear map system encodes a value $a \in \mathbb{Z}_p$ (where p is a large prime) by using a *homomorphic* encryption scheme equipped with some additional structure. In other words, given encodings of a and b, one can perform homomorphic computations by computing encodings of $a + b$ and $a \cdot b$. Additionally, each multilinear map encoding is associated with some *level* described by a value $i \in \{1 \dots \kappa\}$ for a fixed universe parameter κ. Encodings can be added only if they are at the same level: $\mathsf{Enc}_i(a) \oplus \mathsf{Enc}_i(b) \to \mathsf{Enc}_i(a+b)$. Encodings can be multiplied: $\mathsf{Enc}_i(a) \odot \mathsf{Enc}_j(b) \to \mathsf{Enc}_{i+j}(a \cdot b)$ if $i + j \leq \kappa$ but is meaningless otherwise. We naturally extend the encoding procedure and the homomorphic operations to encode and to compute on matrices, respectively, by encoding each term of the matrix separately. Finally, the multilinear map system comes equipped with a *zero test*: an efficient procedure for testing whether the input is an encoding of 0 at level-κ. However, such zero-test procedure is not perfect as desired when instantiated with concrete candidate multilinear maps. In particular we are interested in the imperfection in GGH13 map.

An Imperfection of the GGH13 Multilinear Maps

Expanding a little on the abstraction above, a fresh multilinear map encoding of a value $a \in \mathbb{Z}_p$ at level i is obtained by first sampling a random value μ from $\mathbb{Z}_p$ and then encoding $\mathsf{Enc}_i(a + \mu \cdot p)$. Homomorphic operations can be performed just as before, except that the randomnesses from different encodings also get computed on. Specifically, $\mathsf{Enc}_i(a + \mu \cdot p) \oplus \mathsf{Enc}_i(b + \nu \cdot p)$ yields $\mathsf{Enc}_i(a + b + (\mu + \nu) \cdot p)$ and multiplication $\mathsf{Enc}_i(a + \mu \cdot p) \odot \mathsf{Enc}_j(b + \nu \cdot p)$ yields $\mathsf{Enc}_{i+j}(a \cdot b + (b \cdot \mu + a \cdot \nu + \mu \cdot \nu \cdot p) \cdot p)$ if $i + j \leq \kappa$ but is meaningless otherwise. An imperfection of the zero-test procedure is a feature characterized by two phenomena:

1. On input $\mathsf{Enc}_\kappa(0 + r \cdot p)$ the zero-test procedure additionally reveals r in a somewhat "scrambled" form.
2. For certain efficiently computable polynomials f and a collection of scrambled values $\{r_i\}$ it is efficient to check if $f(\{r_i\}) = 0 \mod p$ or not for any choice of r_i's.[7]

This imperfection has been exploited to perform attacks in prior works, such as the one by Miles et al. [32].[8]

Matrix Branching Programs

A matrix branching program of length ℓ for n-bit inputs is a sequence $BP = \left\{A_0, \left\{A_{i,0}, A_{i,1}\right\}_{i=1}^{\ell}, A_{\ell+1}\right\}$, where $A_0 \in \{0,1\}^{1\times 5}$, $A_{i,b}$'s for $i \in [\ell]$ are in $\{0,1\}^{5\times 5}$ and $A_{\ell+1} \in \{0,1\}^{5\times 1}$. Without providing details, we note that the choice of 5×5 matrices comes from Barrington's Theorem [10]. We use the notation $[n]$ to describe the set $\{1, \dots, n\}$.

[7] One can alternatively consider the scrambled values as polynomials over $\{r_i\}$ and then check if $f(\{r_i\})$ is identically zero in $\mathbb{Z}_p$.

[8] Recent works such as [26, 23], have attempted to realize obfuscation schemes secure against such imperfection and are provably secure against our attacks. We refer to them as obfuscations from weak multilinear maps (see Figure 1).

Let inp be a fixed function such that $\mathsf{inp}(i) \in [n]$ is the input bit position examined in the i^{th} step of the branching program. The function computed by this matrix branching program is

$$f_{BP}(x) = \begin{cases} 0 & \text{if } A_0 \cdot \prod_{i=1}^{\ell} A_{i,x[\mathsf{inp}(i)]} \cdot A_{\ell+1} = 0 \\ 1 & \text{if } A_0 \cdot \prod_{i=1}^{\ell} A_{i,x[\mathsf{inp}(i)]} \cdot A_{\ell+1} \neq 0 \end{cases},$$

where $x[\mathsf{inp}(i)] \in \{0,1\}$ denotes the $\mathsf{inp}(i)^{th}$ bit of x.

The branching program described above inspects one bit of the input in each step. More generally, multi-arity branching programs inspect multiple bits in each step. For example, dual-input programs inspect two bits during each step. Our strategy only works against single-input branching programs, hence we restrict ourselves to that setting.

Exploiting the Imperfection/Weakness

At a high level, obfuscation of a branching program $BP = \{A_0, \{A_{i,0}, A_{i,1}\}_{i=1}^{\ell}, A_{\ell+1}\}$ yields a collection of encodings $\{M_0, \{M_{i,0}, M_{i,1}\}_{i=1}^{\ell}, M_{\ell+1}\}$, say all of which are obtained at level-1.[9] We let $\{Z_0, \{Z_{i,0}, Z_{i,1}\}_{i=1}^{\ell}, Z_{\ell+1}\}$ denote the randomnesses used in the generation of these encodings, where each Z corresponds to a matrix of random values (analogous to r above) in $\mathbb{Z}_p$. For every input x such that $BP(x) = 0$, we have that $M_0 \odot \bigodot_{i=1}^{\ell} M_{i,x[\mathsf{inp}(i)]} \odot M_{\ell+1}$ is an encoding of 0, say of the form $\mathsf{Enc}(0 + r_x \cdot p)$ from which r_x can be learned in a scrambled form. The crucial observations of Miles et al. [32] are: (1) for every known obfuscation construction, r_x is a *program dependent* function of $\{Z_0, \{Z_{i,0}, Z_{i,1}\}_{i=1}^{\ell}, Z_{\ell+1}\}$, and (2) for a large enough $m \in \mathbb{Z}$ the values $\{r_{x_k}\}_{k=1}^{m}$ must be correlated, which in turn implies that there exists a (program-dependent) efficiently computable function f^{BP} and input choices $\{x_k^{BP}\}_{k=1}^{m}$ such that for all $k \in [m]$, $BP(x_k^{BP}) = 0$ and $f^{BP}(\{r_{x_k^{BP}}\}_{k=1}^{m}) = 0 \mod p$.[10] Further, just like Miles et al. we are interested in constructing an attacker for the *indistinguishability* notion of obfuscation. In this case, given two arbitrarily distinct programs BP and BP' (such that $\forall x, BP(x) = BP'(x)$) an attacker needs to distinguish between the obfuscations of BP and BP'. Therefore, to complete the attack, it suffices to argue that for the sequence of $\{r'_{x_k^{BP'}}\}$ values obtained from execution of BP' it holds that, $f^{BP}(\{r'_{x_k^{BP'}}\}_{k=1}^{m}) \neq 0 \mod p$. Hence, the task of attacking any obfuscation scheme reduces to the task of finding such distinguishing function f^{BP}.

Miles et al. [32] accomplishes that by presenting specific examples of branching programs, both of which implement the constant zero function, and a corresponding distinguishing function. They then extend the attack to other related branching programs that are padded with those constant-zero programs. The details of their attack [32] is quite involved, hence we jump directly to the intuition behind our envisioned more general attacks.

Partial Inequivalence of Branching Programs and Our Attacks

We start with the following observation. For BGKPS-like-obfuscations for any branching program $BP = \{A_0, \{A_{i,0}, A_{i,1}\}_{i=1}^{\ell}, A_{\ell+1}\}$ the value $s_x = r_x \mod p$ looks something like: [11]

[9] Many obfuscation constructions use more sophisticate leveling structure, typically referred to as so-called "straddling sets". However we emphasize that, this structure does not affect our attacks. Therefore we will just ignore this in our setting.

[10] This follows from the existence of an annihilating polynomial for any over-determined non-linear systems of equations. We refer to [30] for more details.

[11] Obtaining this expression requires careful analysis that is deferred to the main body of the paper. Also, by abuse of notation let $A_{0,x_{\mathsf{inp}(0)}} = A_0$, $A_{\ell+1,x_{\mathsf{inp}(\ell+1)}} = A_{\ell+1}$, $Z_{0,x_{\mathsf{inp}(0)}} = Z_0$ and $Z_{\ell+1,x_{\mathsf{inp}(\ell+1)}} = Z_{\ell+1}$.

$$s_x \simeq \prod_{i=1}^{\ell} \alpha_{i,x[\mathsf{inp}(i)]} \underbrace{\left[\sum_{i=0}^{\ell+1} \left(\prod_{j=0}^{i-1} A_{j,x_{\mathsf{inp}(j)}} \cdot Z_{i,x[\mathsf{inp}(i)]} \cdot \prod_{j=i+1}^{\ell+1} A_{j,x_{\mathsf{inp}(j)}} \right)\right]}_{t_x},$$

where $\{Z_0, \{Z_{i,0}, Z_{i,1}\}_{i=1}^{\ell}, Z_{\ell+1}\}$ where $\{Z_0, \{Z_{i,0}, Z_{i,1}\}_{i=1}^{\ell}, Z_{\ell+1}\}$ are the randomnesses contributed by the corresponding encodings. Let $\overline{x}$ denote the value obtained by flipping every bit of x (a.k.a. the bitwise complement). Now observe that the product value $\Lambda = \prod_{i=1}^{\ell} \alpha_{i,x[\mathsf{inp}(i)]} \cdot \alpha_{i,\overline{x}[\mathsf{inp}(i)]}$ is independent of x. Therefore, $u_x = s_x \cdot s_{\overline{x}} = \Lambda \cdot t_x \cdot t_{\overline{x}}$. Absorbing Λ in the $\{Z_{i,0}, Z_{i,1}\}_{i=1}^{\ell}$, we have that u_x is quadratic in the randomness values $\{Z_0, \{Z_{i,0}, Z_{i,1}\}_{i=1}^{\ell}, Z_{\ell+1}\}$, or linear in the random terms ZZ' obtained by multiplying every choice of $Z, Z' \in \{Z_0, \{Z_{i,0}, Z_{i,1}\}_{i=1}^{\ell}, Z_{\ell+1}\}$. In other words if BP evaluates to 0 both on inputs x and $\overline{x}$, the values revealed by two zero-test operations give one linear equation where the coefficients of the linear equations are program dependent. Now, if BP implements a "sufficiently non-evasive" circuit,(e.g. a PRF) such that there exist sufficiently many such inputs $x, \overline{x}$ for which $BP(x) = BP(\overline{x}) = 0$, then collecting sufficiently many values $\{x_k^{BP}, u_{x_k^{BP}}\}_{k=1}^{m}$, we get a dependent system of linear relations. Namely, there exist $\{\nu_k^{BP}\}_{k=1}^{m}$ such that $\sum_{k=1}^{m} \nu_k^{BP} \cdot u_{x_k^{BP}} = 0$. In other words, $\sum_{k=1}^{m} \nu_k^{BP} \cdot r_{x_k^{BP}} \cdot r_{\overline{x}_k^{BP}} = 0 \mod p$, where $\{\nu_k^{BP}\}_{k=1}^{m}$ depends only on the description of the branching program BP.

We remark that, in the process of linearization above we increased (by a quadratic factor) the number of random terms in the system. However, this can be always compensated by using more equations, because the number of random terms is $O(poly(n))$ (n is the input length) whereas the number of choices of input x is $2^{O(n)}$ which implies that there are exponentially many r_x available.

Note that for any branching program BP' that is "different enough" from BP, we could expect that $\sum_{k=1}^{m} \nu_k^{BP} \cdot r'_{x_k^{BP}} \cdot r'_{\overline{x}_k^{BP}} \neq 0 \mod p$ where $r'_{x_k^{BP}}$ are values revealed in executions of an obfuscation of BP'. This is because the values $\{\nu_k^{BP}\}_{k=1}^{m}$ depend on the specific implementation of BP through terms of the form $\prod_{j=0}^{i-1} A_{j,x[\mathsf{inp}(i)]}$ and $\prod_{j=i+1}^{\ell+1} A_{j,x[\mathsf{inp}(i)]}$ in s_x above. Two branching programs that differ from each other in this sense are referred to as *partially inequivalent*.[12]

What Programs are Partially Inequivalent? Attack on NC^1 circuits

The condition we put forth seems to be fairly generic and intuitively should work for large class of programs. In particular, we are interested in the programs generated from NC^1 circuits. However, due to complex structures of such programs the analysis becomes quite non-trivial.[13] Nonetheless, we manage to show via implementation in Sage [35] that the attack indeed works on a pair of branching programs obtained from a pair of simple NC^1 circuits, (say C_0, C_1) (see Sec. 6 for the circuit descriptions) by applying Barrington's Theorem. The circuits take 4 bits of inputs and on any input they evaluate to 0. In our attack we use

[12] Note that the only other constraint we need is that both BP and BP' evaluates to 0 for sufficiently many inputs, which we include in the definition (c.f. Def. 2) of partial inequivalence.

[13] Note that, the analysis of Miles et al. uses 2×2 matrices in addition to using simple branching programs. These simplifications allow them to base their analysis on many facts related to the structures of these programs. Our aim here is to see if the attack works for programs obtained from NC^1 circuits, in particular via Barrington's Theorem. So, unfortunately it is not clear if their approach can be applicable here as the structure of the programs yielded via Barrington's Theorem become much complex structurally (and also much larger in size) to analyze.

all possible 16 inputs. Furthermore, we can escalate the attack to any pair of NC^1 circuits (E_0, E_1) where $E_b = \neg C_b \wedge D_b$ $(b \in \{0,1\})$ for practically any two NC^1 circuits D_0, D_1 (we need only one input x for which $D(x) = D(\overline{x}) = 0$). We now take again a sequence of 16-inputs such that we vary the parts of all the inputs going into C_b and keep the part of inputs read by D_b fixed to x. Intuitively, since the input to D_b is always the same, each evaluation chooses the exactly same randomnesses (that is Z_i's) always. Hence in the resulting system all the random variables can be replaced by a single random variable and hence $\neg C_b \wedge D_b$ can be effectively "collapsed" to a much smaller circuit $\neg C_b \wedge 0$ (0 refers to the smallest trivial circuit consisting of only identities). Finally, again via our Sage-implementation we show that for circuits $\neg C_0 \wedge 0$ and $\neg C_1 \wedge 0$ the corresponding branching programs are partially inequivalent.

As a corollary we are also able to show examples of universal circuits U_b for which the same attack works. Since the circuit D can be almost any arbitrary NC^1 circuit, we can, in particular use any universal circuit U' and carefully combine that with C to obtain our attackable universal circuit U that results in partially inequivalent Barrington programs.

2 Notations and Preliminaries

2.1 Notations

We denote the set of natural numbers $\{1, 2, \ldots\}$ by $\mathbb{N}$, the set of all integers $\{\ldots, -1, 0, 1 \ldots\}$ by $\mathbb{Z}$ and the set of real numbers by $\mathbb{R}$. We use the notation $[n]$ to denote the set of first n natural numbers, namely $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$.

For any bit-string $x \in \{0,1\}^n$ we let $x[i]$ denotes the i-th bit. For a matrix A we denote its i-th row by $A[i, \star]$, its j-th column by $A[\star, j]$ and the element in the i-th row and j-th column by $A[i, j]$. The i-th element of a vector $\boldsymbol{v}$ is denoted by $\boldsymbol{v}[i]$.

For more notational conventions we refer to the full version [4].

3 Attack Model for Investigating Annihilation Attacks

Similar to Miles, Sahai, and Zhandry [32] we use an abstract attack model designed to encompass the main ideas of BGKPS-like-obfuscations [8, 12, 3, 33, 31, 7] as the starting point for our attacks. We formally describe the model in the full version [4].

4 Partially Inequivalent Branching Programs

In this section, we provide a formal condition on two single-input branching programs (naturally extends to multi-input settings), namely partial inequivalence, that is sufficient for launching a distinguishing attack in the abstract model. In Section 5 we prove that this condition is sufficient for the attack.[14]

▶ **Definition 1** (Partial Products)**.** Let $\mathbf{A} = (\mathsf{inp}, A_0, \{A_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, A_{\ell+1})$ be a single-input branching program of matrix-dimension d and length ℓ over n-bit input.

[14] We note that this condition is not necessary. Looking ahead, we only consider first order partially inequivalent programs in paper and remark that higher order partially inequivalent programs could also be distinguished using our techniques.

1. For any input $x \in \{0,1\}^n$ and any index $i \in [\ell+1] \cup \{0\}$ we define the vectors $\phi^{(i)}_{\mathbf{A},x}$ as follows:

$$\phi^{(i)}_{\mathbf{A},x} \stackrel{\text{def}}{=} \begin{cases} \left(A_0 \cdot \prod_{j=1}^{i-1} A_{j,x[\mathsf{inp}(j)]}\right) \otimes \left(\prod_{j=i+1}^{\ell} A_{j,x[\mathsf{inp}(j)]} \cdot A_{\ell+1}\right)^T \in \{0,1\}^{1\times d^2} & \text{if } i \in [\ell]\,, \\ \left(\prod_{j=1}^{\ell} A_{j,x[\mathsf{inp}(j)]} \cdot A_{\ell+1}\right)^T \in \{0,1\}^{1\times d} & \text{if } i = 0\,, \\ A_0 \cdot \prod_{j=1}^{\ell} A_{j,x[\mathsf{inp}(j)]} \in \{0,1\}^{1\times d} & \text{if } i = \ell+1\,. \end{cases}$$

   Additionally, define $\widetilde{\phi}^{(i)}_{\mathbf{A},x}$ for any such branching program as:

$$\widetilde{\phi}^{(i)}_{\mathbf{A},x} \stackrel{\text{def}}{=} \begin{cases} [\phi^{(i)}_{\mathbf{A},x} \mid 0^{d^2}] & \text{if } i \in [\ell] \text{ and } x[\mathsf{inp}(i)] = 0\,, \\ [0^{d^2} \mid \phi^{(i)}_{\mathbf{A},x}] & \text{if } i \in [\ell] \text{ and } x[\mathsf{inp}(i)] = 1\,, \\ \phi^{(i)}_{\mathbf{A},x} & \text{if } i = 0 \text{ or } \ell+1\,, \end{cases}$$

   where inp is a function from $[\ell] \to [n]$ and that $x[\mathsf{inp}(i)]$ denotes the bit of x corresponding to location described by $\mathsf{inp}(x)$.
2. Then the **linear partial product vector** $\boldsymbol{\phi}_{\mathbf{A},x}$ and the **quadratic partial product vector** $\boldsymbol{\psi}_{\mathbf{A},x}$ of $\mathbf{A}$ with respect to x are defined as:

$$\boldsymbol{\phi}_{\mathbf{A},x} \stackrel{\text{def}}{=} [\widetilde{\phi}^{(0)}_{\mathbf{A},x} \mid \cdots \mid \widetilde{\phi}^{(\ell+1)}_{\mathbf{A},x}] \in \{0,1\}^{1\times(2d+2^\ell d^2)}\,,$$
$$\boldsymbol{\psi}_{\mathbf{A},x} \stackrel{\text{def}}{=} \boldsymbol{\phi}_{\mathbf{A},x} \otimes \boldsymbol{\phi}_{\mathbf{A},\overline{x}} \in \{0,1\}^{1\times(2d+2^\ell d^2)^2}\,,$$

   where $\overline{x} = x \oplus 1^n$ is the compliment of x.
3. For a set of inputs $X = \{x_1, x_2, \ldots, x_m\}$ the the **linear partial product matrix** $\Phi_{\mathbf{A},X}$ and the **quadratic partial product matrix** $\Psi_{\mathbf{A},X}$ of $\mathbf{A}$ with respect to X are defined as:

$$\Phi_{\mathbf{A},X} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{\phi}_{\mathbf{A},x_1} \\ \boldsymbol{\phi}_{\mathbf{A},x_2} \\ \vdots \\ \boldsymbol{\phi}_{\mathbf{A},x_m} \end{bmatrix} \in \{0,1\}^{m\times(2d+2^\ell d^2)}\,,$$

$$\Psi_{\mathbf{A},X} \stackrel{\text{def}}{=} \Phi_{\mathbf{A},X} \boxtimes \Phi_{\mathbf{A},\overline{X}} + \Phi_{\mathbf{A},\overline{X}} \boxtimes \Phi_{\mathbf{A},X} = \begin{bmatrix} \boldsymbol{\psi}_{\mathbf{A},x_1} + \boldsymbol{\psi}_{\mathbf{A},\overline{x}_1} \\ \boldsymbol{\psi}_{\mathbf{A},x_2} + \boldsymbol{\psi}_{\mathbf{A},\overline{x}_2} \\ \vdots \\ \boldsymbol{\psi}_{\mathbf{A},x_m} + \boldsymbol{\psi}_{\mathbf{A},\overline{x}_m} \end{bmatrix} \in \{0,1\}^{m\times(2d+2^\ell d^2)^2},$$

   where $\overline{X} \stackrel{\text{def}}{=} \{\overline{x}_1, \overline{x}_2, \ldots\}$.

▶ **Definition 2** (Partial Inequivalence)**.** Let $\mathbf{A}_0$ and $\mathbf{A}_1$ be two single-input matrix branching programs of matrix-dimension d and length ℓ over n-bit input. Then they are called **partially inequivalent** if there exists a polynomial in security parameter sized set X of inputs such that:

- For every $x \in X$, we have that $\mathbf{A}_0(x) = \mathbf{A}_1(x) = 0$ and $\mathbf{A}_0(\overline{x}) = \mathbf{A}_1(\overline{x}) = 0$.
- $\mathsf{colsp}\left(\Psi_{\mathbf{A}_0,X}\right) \neq \mathsf{colsp}\left(\Psi_{\mathbf{A}_1,X}\right)$.

5 Annihilation Attacks for Partially Inequivalent Programs

In this section, we describe an abstract annihilation attack against any two branching programs that are partially inequivalent. We show an attack only in the abstract model and provide details on how it can be extended to the real GGH13 setting in the full version.

▶ **Theorem 3.** *Let $\mathcal{O}$ be the generic obfuscator described in Section 3.2 of the full version. Then for any two functionally equivalent same length single-input branching programs $\mathbf{A}_0, \mathbf{A}_1$ that are partially inequivalent there exists a probabilistic polynomial time attacker that distinguishes between between $\mathcal{O}(\mathbf{A}_0)$ and $\mathcal{O}(\mathbf{A}_1)$ with noticeable probability in the abstract attack model.*

Proof.

Setup for the attack

The given branching programs $\mathbf{A}_0$ and $\mathbf{A}_1$ are provided to be functionally equivalent and partially inequivalent. Therefore there exists a set X such that: (1) for all $x \in X, \mathbf{A}_0(x) = \mathbf{A}_0(\overline{x}) = \mathbf{A}_1(x) = \mathbf{A}_1(\overline{x}) = 0$, and (2) $\mathsf{colsp}\left(\Psi_{\mathbf{A}_0,X}\right) \neq \mathsf{colsp}\left(\Psi_{\mathbf{A}_1,X}\right)$. We will assume that the adversary has access to X as auxiliary information.

Challenge

$\mathcal{A}$ receives as a challenge the obfuscation of the branching program: $\mathbf{A} \in \{\mathbf{A}_0, \mathbf{A}_1\}$ by the challenger. Recall from the description of the abstract obfuscator that, the obfuscation of program $\mathbf{A} = (\mathsf{inp}, A_0, \{A_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, A_{\ell+1})$, denoted by $\mathcal{O}(\mathbf{A})$ consists of the following public variables:

$$Y_0 := A_0 \cdot R_1^{adj} + gZ_0, \quad Y_{i,b} := \alpha_{i,b} R_i \cdot A_{i,b} \cdot R_{i+1}^{adj} + gZ_{i,b}, \quad Y_0 := R_{\ell+1} \cdot A_{\ell+1} + gZ_0,$$

where the arbitrary secret variables are:

$$\widetilde{A}_0 \stackrel{\text{def}}{=} A_0 \cdot R_1^{adj}, \quad \widetilde{A}_{i,b} \stackrel{\text{def}}{=} \alpha_{i,b}(R_{i,b} \cdot A_{i,b} \cdot R_{i,b}^{adj}), \quad \widetilde{A}_{\ell+1} \stackrel{\text{def}}{=} R_{\ell+1} \cdot A_{\ell+1};$$

for random variables (i.e. Killian randomizers) $R_1, \{R_i\}, R_{\ell+1}$ and the random secret variables are denoted by $Z_0, \{Z_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, Z_{\ell+1}$ and the special secret variable is g. Via change of variables we can equivalently write:

$$Y_0 := (A_0 + gZ_0) \cdot R_1^{adj}; \quad Y_{i,b} := \alpha_{i,b} R_i \cdot (A_{i,b} + gZ_{i,b}) \cdot R_{i+1}^{adj}; \quad Y_{\ell+1} := R_{\ell+1} \cdot (A_{\ell+1} + gZ_{\ell+1}).$$

Pre-Zeroizing Computation (Type-1 queries)

On receiving the obfuscation of $\mathbf{A} \in \{\mathbf{A}_0, \mathbf{A}_1\}$, $\mathcal{O}(\mathbf{A}) = \{Y_0, \{Y_{i,b}\}, Y_{\ell+1}\}$ the attacker, in the pre-zeroizing step, performs a "valid" Type-1 queries on all the inputs $X, \overline{X}$ where $X = \{x_1, \ldots, x_m\}, \overline{X} = \{\overline{x}_1, \ldots, \overline{x}_m\}$. That is, for an $x \in \{0,1\}^n$, and the abstract obfuscation $\mathcal{O}(\mathbf{A})$, the attacker queries the polynomial:

$$P_{\mathbf{A},x} = Y_0 \cdot \prod_{i=1}^{\ell} Y_{i,x[\mathsf{inp}(i)]} \cdot Y_{\ell+1}.$$

Then, expressing $P_{\mathbf{A},x}$ stratified as powers of g we obtain:

$$P_{\mathbf{A},x} = P_{\mathbf{A},x}^{(0)}(\{Y_i\}_i) + g \cdot P_{\mathbf{A},x}^{(1)}(\{Y_i\}_i) + \ldots + g^{\ell+2} \cdot P_{\mathbf{A},x}^{(\ell+2)}(\{Y_i\}_i)$$

for some polynomials $P^{(j)}_{\mathbf{A},x}(\{Y_i\}_i)$ ($j \in \{0,...,\ell+1\}$). However, by Lemma 4 we have that:

$$P^{(0)}_{\mathbf{A},x} = \rho\widehat{\alpha}_x\mathbf{A}(x)$$

for $\rho \stackrel{\text{def}}{=} \prod_i \det(R_i)$ (or $\rho I = \prod_i R_i^{adj} R_i$) and $\widehat{\alpha}_x \stackrel{\text{def}}{=} \prod_{i=1}^{\ell} \alpha_{i,x_{\mathsf{inp}(i)}}$. Since for $x \in X$ we have that $\mathbf{A}(x) = 0$, the polynomial $P^{(0)}_{\mathbf{A},x}$ is identically 0. Consequently, for each such Type 1 query the attacker receives a new handle to a variable $W_{\mathbf{A},x}$ that can be expressed as follows:

$$W_{\mathbf{A},x} = P_{\mathbf{A},x}/g = P^{(1)}_{\mathbf{A},x} + g \cdot P^{(2)}_{\mathbf{A},x} + ... + g^{\ell+1} \cdot P^{(\ell+2)}_{\mathbf{A},x}.$$

Analogously, the attacker obtains handles $W_{\mathbf{A},\overline{x}}$. After obtaining handles

$$\{(W_{\mathbf{A},x_1}, W_{\mathbf{A},\overline{x}_1}), ...(W_{\mathbf{A},x_m}, W_{\mathbf{A},\overline{x}_m})\}$$

the attacker starts the post-zeroizing phase.

Post-Zeroizing Computation

The goal of post-zeroizing computation is to *find* a polynomial Q^{ann} of degree $poly(\lambda)$ such that following holds for some $b \in \{0,1\}$:

(i) $Q^{\mathsf{ann}}(P^{(1)}_{\mathbf{A}_b,x_1}, P^{(1)}_{\mathbf{A}_b,\overline{x}_1}..., P^{(1)}_{\mathbf{A}_b,x_m}, P^{(1)}_{\mathbf{A}_b,\overline{x}_m}) \equiv 0.$

(ii) $Q^{\mathsf{ann}}(P^{(1)}_{\mathbf{A}_{1-b},x_1}, P^{(1)}_{\mathbf{A}_{1-b},\overline{x}_1}..., P^{(1)}_{\mathbf{A}_{1-b},x_m}, P^{(1)}_{\mathbf{A}_{1-b},\overline{x}_m}) \not\equiv 0.$

Clearly, this leads to an attack on the obfuscation security as $\mathcal{A}$ would receive 0 from the challenger if and only if $Q^{\mathsf{ann}}(P^{(1)}_{\mathbf{A},x_1}, P^{(1)}_{\mathbf{A},\overline{x}_1}..., P^{(1)}_{\mathbf{A},x_m}, P^{(1)}_{\mathbf{A},\overline{x}_m})$ is identically zero, hence it would receive 0 if and only if $\mathbf{A}_b$ is chosen by the challenger in the challenge phase. To find such Q^{ann} the attacker continues as follows. Observe that by Lemma 4, for every $x \in X$ we have that:

$$P^{(1)}_{\mathbf{A},x} = \rho\widehat{\alpha}_x(\boldsymbol{\phi}_{\mathbf{A},x} \cdot \boldsymbol{z}^T)', \tag{1}$$

$$P^{(1)}_{\mathbf{A},\overline{x}} = \rho\widehat{\alpha}_{\overline{x}}(\boldsymbol{\phi}_{\mathbf{A},\overline{x}} \cdot \boldsymbol{z}^T). \tag{2}$$

Next, multiplying the polynomials $P^{(1)}_{\mathbf{A},x}$ and $P^{(1)}_{\mathbf{A},\overline{x}}$ (Eq. 1 and Eq. 2) we get:

$$\widetilde{P}^{(1)}_{\mathbf{A},x} \stackrel{\text{def}}{=} P^{(1)}_{\mathbf{A},x}P^{(1)}_{\mathbf{A},\overline{x}} = \rho^2\widehat{\alpha}\left((\boldsymbol{\phi}_{\mathbf{A},x} \cdot \boldsymbol{z}) \otimes (\boldsymbol{\phi}_{\mathbf{A},\overline{x}} \cdot \boldsymbol{z})\right) \tag{3}$$

$$= \rho^2\widehat{\alpha}\left((\boldsymbol{\phi}_{\mathbf{A},x} \otimes \boldsymbol{\phi}_{\mathbf{A},\overline{x}}) \cdot (\boldsymbol{z}^T \otimes \boldsymbol{z}^T)\right) \tag{4}$$

$$= \rho^2\widehat{\alpha}(\boldsymbol{\psi}_{\mathbf{A},x} \cdot \boldsymbol{z}^T \otimes \boldsymbol{z}^T).$$

where $\widehat{\alpha} \stackrel{\text{def}}{=} \widehat{\alpha}_x\widehat{\alpha}_{\overline{x}}$ is now independent of input x.[15] Similarly we can also have:

$$\widetilde{P}^{(1)}_{\mathbf{A},\overline{x}} \stackrel{\text{def}}{=} P^{(1)}_{\mathbf{A},\overline{x}}P^{(1)}_{\mathbf{A},x} = \rho^2\widehat{\alpha}\left((\boldsymbol{\phi}_{\mathbf{A},\overline{x}} \cdot \boldsymbol{z}) \otimes (\boldsymbol{\phi}_{\mathbf{A},x} \cdot \boldsymbol{z})\right)$$

$$= \rho^2\widehat{\alpha}\left((\boldsymbol{\phi}_{\mathbf{A},\overline{x}} \otimes \boldsymbol{\phi}_{\mathbf{A},x}) \cdot (\boldsymbol{z}^T \otimes \boldsymbol{z}^T)\right)$$

$$= \rho^2\widehat{\alpha}(\boldsymbol{\psi}_{\mathbf{A},\overline{x}} \cdot \boldsymbol{z}^T \otimes \boldsymbol{z}^T).$$

[15] Here, we use the fact that the branching programs are single-input. For multi-input programs we do *not* know how to make $\widehat{\alpha}$ independent of x. The rest of the analysis does not require the programs to be single-input.

However, since field multiplication is commutative, adding we get:

$$\begin{aligned}\widetilde{P}^{(1)}_{\mathbf{A},x} + \widetilde{P}^{(1)}_{\mathbf{A},\overline{x}} = 2P^{(1)}_{\mathbf{A},x}P^{(1)}_{\mathbf{A},\overline{x}} &= \rho^2\widehat{\alpha}(\boldsymbol{\psi}_{\mathbf{A},x}\cdot \boldsymbol{z}^T \otimes \boldsymbol{z}^T) + \rho^2\widehat{\alpha}(\boldsymbol{\psi}_{\mathbf{A},\overline{x}}\cdot \boldsymbol{z}^T \otimes \boldsymbol{z}^T)\\ &= \rho^2\widehat{\alpha}(\boldsymbol{\psi}_{\mathbf{A},x} + \boldsymbol{\psi}_{\mathbf{A},\overline{x}})\cdot(\boldsymbol{z}^T \otimes \boldsymbol{z}^T)\,.\end{aligned}$$

Using the given conditions that $\Psi_{\mathbf{A}_0,X}$ and $\Psi_{\mathbf{A}_1,X}$ have distinct column spaces (and hence distinct left-kernel) the attacker can efficiently compute (e.g. via Gaussian Elimination) a vector $\boldsymbol{v}_{\mathsf{ann}} \in \{0,1\}^{1\times m}$ that belongs to its left-kernel, call it the *annihilating vector*, such that for some $b \in \{0,1\}$ we have:

$$\boldsymbol{v}_{\mathsf{ann}}\cdot\Psi_{\mathbf{A}_b,X} = 0 \qquad \text{but} \qquad \boldsymbol{v}_{\mathsf{ann}}\cdot\Psi_{\mathbf{A}_{1-b},X} \neq 0.$$

The corresponding annihilation polynomial Q^{ann} can be written as:

$$Q^{\mathsf{ann}}_{\boldsymbol{v}_{\mathsf{ann}}}(W_{\mathbf{A},x_1}, W_{\mathbf{A},\overline{x}_1}, \ldots, W_{\mathbf{A},x_m}, W_{\mathbf{A},\overline{x}_m}) = \boldsymbol{v}_{\mathsf{ann}}\cdot\begin{bmatrix} W_{\mathbf{A},x_1}W_{\mathbf{A},\overline{x}_1}\\ \vdots \\ W_{\mathbf{A},x_m}W_{\mathbf{A},\overline{x}_m}\end{bmatrix}.$$

Observe that the coefficient of g^0 in the expression $Q^{\mathsf{ann}}_{\boldsymbol{v}_{\mathsf{ann}}}(W_{\mathbf{A},x_1}, W_{\mathbf{A},\overline{x}_1}, \ldots, W_{\mathbf{A},x_m}, W_{\mathbf{A},\overline{x}_m})$ from above is equal to $Q^{\mathsf{ann}}_{\boldsymbol{v}_{\mathsf{ann}}}(P^{(1)}_{\mathbf{A}_b,x_1}, P^{(1)}_{\mathbf{A}_b,\overline{x}_1}..., P^{(1)}_{\mathbf{A}_b,x_m}, P^{(1)}_{\mathbf{A}_b,\overline{x}_m})$. Moreover this value for $\mathbf{A} = \mathbf{A}_b$ is:

$$Q^{\mathsf{ann}}_{\boldsymbol{v}_{\mathsf{ann}}}(P^{(1)}_{\mathbf{A}_b,x_1}, P^{(1)}_{\mathbf{A}_b,\overline{x}_1}..., P^{(1)}_{\mathbf{A}_b,x_m}, P^{(1)}_{\mathbf{A}_b,\overline{x}_m}) = \boldsymbol{v}_{\mathsf{ann}}\cdot\frac{\Psi_{\mathbf{A}_b,X}}{2}\cdot(\boldsymbol{z}\otimes\boldsymbol{z})^T \equiv 0$$

but for $\mathbf{A}_{1-b}$:

$$Q^{\mathsf{ann}}_{\boldsymbol{v}_{\mathsf{ann}}}(P^{(1)}_{\mathbf{A}_{1-b},x_1}, P^{(1)}_{\mathbf{A}_{1-b},\overline{x}_1}..., P^{(1)}_{\mathbf{A}_{1-b},x_m}, P^{(1)}_{\mathbf{A}_{1-b},\overline{x}_m}) = \boldsymbol{v}_{\mathsf{ann}}\cdot\frac{\Psi_{\mathbf{A}_{1-b},X}}{2}\cdot(\boldsymbol{z}\otimes\boldsymbol{z})^T \not\equiv 0.$$

Hence, the response to Type 2 query is sufficient to distinguish between obfuscation of $\mathbf{A}_b$ and $\mathbf{A}_{1-b}$ in the abstract model. This concludes the proof. ◀

Evaluations of $P^{(0)}_{\mathbf{A},\boldsymbol{x}}$ and $P^{(1)}_{\mathbf{A},\boldsymbol{x}}$

Below we state a lemma without proof (that is deferred to the full version) that described what the terms $P^{(0)}_{\mathbf{A},x}$ and $P^{(1)}_{\mathbf{A},x}$ look like.

▶ **Lemma 4.** *For every $x \in \{0,1\}^n$, we have that:*

$$P^{(0)}_{\mathbf{A},x} = \rho\widehat{\alpha}_x\mathbf{A}(x)\,,$$
$$P^{(1)}_{\mathbf{A},x} = \rho\widehat{\alpha}_x(\boldsymbol{\phi}_{\mathbf{A},x}\cdot\boldsymbol{z}^T),$$

where $\rho \stackrel{\text{def}}{=} \prod_i \det(R_i)$ and $\widehat{\alpha}_x \stackrel{\text{def}}{=} \prod_{i=1}^{\ell}\alpha_{i,x_{\mathsf{inp}(i)}}$ and $\boldsymbol{z}$ is a vector consisting of the random terms $Z_0, Z_{i,b}$, and $Z_{\ell+1}$ used to generate the obfuscation terms $Y_0, Y_{i,b}$, and $Y_{\ell+1}$ in an appropriate sequence.

Extending the Abstract Attack to GGH13 Multilinear Maps

Based on the ideas from Miles et al. we can extend our abstract attacks to actual instantiations with GGH13, that we defer to the full version [4].

6 Example of Partially Inequivalent Circuits

In this section, we show examples of pairs of NC^1 circuits such that the corresponding Barrington-implemented[16] branching programs are partially inequivalent and therefore are subject to the abstract annihilation attacks shown in Section 5. Note that here we extend the notion of partial inequivalence from branching programs to circuits in a natural way. Unless otherwise mentioned, partial inequivalence of circuits specifically imply that the corresponding branching programs generated via applying Barrington's Theorem are partially inequivalent.

6.1 Simple Pairs of Circuits that are Partially Inequivalent

Consider the following pair of circuits (C_0, C_1) each of which implements a boolean function $\{0,1\}^4 \to \{0,1\}$:

$$C_0(x) \stackrel{\text{def}}{=} (x[1] \wedge 1) \bigwedge (x[2] \wedge 0) \bigwedge (x[3] \wedge 1) \bigwedge (x[4] \wedge 0),$$

$$C_1(x) \stackrel{\text{def}}{=} (x[1] \wedge 0) \bigwedge (x[2] \wedge 0) \bigwedge (x[3] \wedge 0) \bigwedge (x[4] \wedge 0).$$

Define the set $X \stackrel{\text{def}}{=} \{0,1\}^4$. Now, we provide an implementation (see the full version [4] for more details on the implementation) in Sage [35] that evaluates the column spaces of matrices produced via applying a Barrington-implementation to the above circuits. The outcome from the implementation led us to conclude the following claim:

▶ **Claim 5.** *Let $\mathbf{A}_{C_0}, \mathbf{A}_{C_1}$ be the Barrington-Implementation of the circuits C_0, C_1 respectively, then we have that:* $\mathsf{colsp}\left(\Psi_{\mathbf{A}_{C_0},X}\right) \neq \mathsf{colsp}\left(\Psi_{\mathbf{A}_{C_1},X}\right)$.

▶ Remark. We emphasize that we use branching programs generated with a particular Barrington-implementation that makes a set of specific choices. We refer the reader to the full version [4] for the details of our implementation. Throughout this section we refer to this particular Barrington-implementation.

The circuits presented above are of constant size. Looking ahead, though, they are partially inequivalent and hence (by Theorem 3) are susceptible to the abstract attack that does not translate to a real-world attack in GGH13 setting immediately. For that we need to consider larger (albeit NC^1) circuits which we construct next based on the above circuits.

6.2 Larger Pairs of Circuits that are Partially Inequivalent

Consider *any* pair of functionally equivalent NC^1 circuits (D_0, D_1) and an input $x^\star \in \{0,1\}^n$ such that $D_0(x^\star) = D_1(x^\star) = D_0(\overline{x^\star}) = D_1(\overline{x^\star}) = 0$. Now define the circuits E_0, E_1 each of which computes a boolean function $\{0,1\}^{n+4} \to \{0,1\}$ as follows:

$$E_0(y) \stackrel{\text{def}}{=} \neg C_0(x) \wedge D_0(x'),$$

$$E_1(y) \stackrel{\text{def}}{=} \neg C_1(x) \wedge D_1(x'),$$

($\neg C$ is the circuit C with output negated) such that for each $y \in \{0,1\}^{n+4}$ we have $y = x \circ x'$ ($\circ$ denotes concatenation) where $x \in \{0,1\}^4$ and $x' \in \{0,1\}^n$. Define the input-sequence $Y \stackrel{\text{def}}{=} \{x \circ x^\star \mid x \in \{0,1\}^4\}$ (consisting of 16 inputs). Then we show the following statement.

[16] Recall that by Barrington-implementation of a circuit we mean the single-input branching program produced as a result of Barrington Theorem on the circuit. Also we implicitly assume that the branching programs are input-oblivious.

▶ **Lemma 6.** *Let* $\mathbf{A}_{E_0}, \mathbf{A}_{E_1}$ *be the Barrington-implementations of* E_0, E_1 *respectively, then we have that:* $\mathsf{colsp}\left(\Psi_{\mathbf{A}_{E_0},Y}\right) \neq \mathsf{colsp}\left(\Psi_{\mathbf{A}_{E_1},Y}\right)$.

6.3 Partially Inequivalent Universal Circuits

In this section we present constructions of (NC^1) universal circuits that, when compiled with two arbitrary distinct (NC^1) but functionally equivalent circuits as inputs, then the obfuscations of the Barrington-implementation of the compiled circuits are distinguishable by the abstract attack.

▶ **Theorem 7.** *There exists a family of* NC^1 *universal circuits* $\mathcal{U} = \{U_1, U_2, \ldots, U_v\}$ *of size* $v = O(poly(\lambda))$ *such that: given two arbitrary functionally equivalent* NC^1 *circuits* G_0, G_1 *that computes arbitrary boolean function* $\{0,1\}^n \to \{0,1\}$ *satisfying (i)* $|G_0| = |G_1| = v$ *and (ii) there exists an input* $x^\star$ *such that* $G_0(x^\star) = G_1(x^\star) = G_0(\overline{x^\star}) = G_1(\overline{x^\star}) = 0$*; then for at least one* $i \in [v]$ *the Barrington-implementations of the circuits* $U_i[G_0]$ *and* $U_i[G_1]$ *are partially inequivalent.*

References

1. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions – cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 153–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53018-4_6.
2. Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 491–520, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53008-5_17.
3. Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 646–658, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press.
4. Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. Cryptology ePrint Archive, Report 2016/1003, 2016. URL: http://eprint.iacr.org/2016/1003.
5. Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 162–172, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-45608-8_9.
6. Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 528–556, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46497-7_21.
7. Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *Advances in Cryptology – EUROCRYPT*, 2016.

8 Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_13.

9 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

10 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5, 1986. doi:10.1145/12130.12131.

11 Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 439–448, Portland, OR, USA, June 14–17, 2015. ACM Press.

12 Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_1.

13 Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. Cryptology ePrint Archive, Report 2016/998, To appear in EUROCRYPT 2017, 2016. URL: http://eprint.iacr.org/2016/998.

14 Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. Personal Communication, 2016.

15 Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new CLT multilinear map over the integers. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 509–536, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_20.

16 Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3–12, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46800-5_1.

17 Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without an encoding of zero. *IACR Cryptology ePrint Archive*, 2016:139, 2016. URL: http://eprint.iacr.org/2016/139.

18 Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *Advances in Cryptology – CRYPTO 2015 – 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 247–266, 2015.

19 Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Com-*

puter Science, pages 607–628, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53008-5_21.

20 Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. Cryptology ePrint Archive, Report 2016/1011, To appear in PKC 2017, 2016. URL: http://eprint.iacr.org/2016/1011.

21 Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_26.

22 Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 559–585, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_20.

23 Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee. Obfuscation from low noise multilinear maps. Cryptology ePrint Archive, Report 2016/599, 2016. http://eprint.iacr.org/2016/599.

24 Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-38348-9_1.

25 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

26 Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *TCC 2016-B*, 2016.

27 Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46497-7_20.

28 Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 308–326, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

29 Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 537–565, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_21.

30 N. Kayal. The complexity of the annihilating polynomial. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*, pages 184–193, July 2009. doi:10.1109/CCC.2009.37.

31 Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. Cryptology ePrint Archive, Report 2014/878, 2014. URL: http://eprint.iacr.org/2014/878.

32 Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 629–658, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53008-5_22.

33 Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_28.

34 Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

35 W. A. Stein et al. *Sage Mathematics Software (Version 7.3)*. The Sage Development Team, 2016. URL: http://www.sagemath.org.

Non-Uniform Attacks Against Pseudoentropy*

Krzysztof Pietrzak[†1] **and Maciej Skorski**[‡2]

1 **Institute of Science and Technology Austria, Klosterneuburg, Austria**
pietrzak@ist.ac.at
2 **Institute of Science and Technology Austria, Klosterneuburg, Austria**
mskorski@ist.ac.at

Abstract

De, Trevisan and Tulsiani [CRYPTO 2010] show that every distribution over n-bit strings which has constant statistical distance to uniform (e.g., the output of a pseudorandom generator mapping $n-1$ to n bit strings), can be distinguished from the uniform distribution with advantage ϵ by a circuit of size $O(2^n\epsilon^2)$.

We generalize this result, showing that a distribution which has less than k bits of min-entropy, can be distinguished from any distribution with k bits of δ-smooth min-entropy with advantage ϵ by a circuit of size $O(2^k\epsilon^2/\delta^2)$. As a special case, this implies that any distribution with support at most 2^k (e.g., the output of a pseudoentropy generator mapping k to n bit strings) can be distinguished from any given distribution with min-entropy $k+1$ with advantage ϵ by a circuit of size $O(2^k\epsilon^2)$.

Our result thus shows that pseudoentropy distributions face basically the same non-uniform attacks as pseudorandom distributions.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases pseudoentropy, non-uniform attacks

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.39

1 Introduction

De, Trevisan and Tulsiani [2] show a non-uniform attack against any pseudorandom generator (PRG) which maps $\{0,1\}^{n-1} \to \{0,1\}^n$. For any $\epsilon \geq 2^{-n/2}$, their attack achieves distinguishing advantage ϵ and can be realized by a circuit of size $O\left(2^n\epsilon^2\right)$. Their attack doesn't even need the PRG to be efficiently computable.

In this work we consider a more general question, where we ask for attacks distinguishing a distribution from any distribution with slightly higher min-entropy. We generalize [2], showing a non-uniform attack which, for any $\epsilon, \delta > 0$, distinguishes any distribution with $< k$ bits of min-entropy from any distribution with k bits of δ-smooth min-entropy with advantage ϵ, and where the distinguisher is of size $O(2^k\epsilon^2/\delta^2)$. As a corollary we recover the [2] result, showing that the output of any pseudoentropy generator $\{0,1\}^k \to \{0,1\}^n$ can be distinguished from any variable with min-entropy $k+1$ with advantage ϵ by circuits of size $O(2^k\epsilon^2)$.

- From a theoretical perspective, we prove where the separation between pseudoentropy and smooth min-entropy lies, by classifying how powerful computationally bounded adversaries can be so they can still be fooled to "see" more entropy than there really is.

* The full version is available at https://arxiv.org/abs/1704.08678
† Supported by the European Research Council, ERC consolidator grant (682815 – TOCNeT).
‡ Supported by the European Research Council, ERC consolidator grant (682815 – TOCNeT).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 39; pp. 39:1–39:13


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

- From a more practical perspective, our result shows that using pseudoentropy instead of pseudorandomness (which for many applications is sufficient and allows for saving in entropy *quantity* [3]), will not give improvements in terms of *quality* (i.e., the size and advantage of distinguishers considered), at least not against generic non-uniform attacks.

1.1 Notation and Basic Definitions

Two variables X and Y are (s,ϵ) indistinguishable, denoted $X \sim_{s,\epsilon} Y$, if for all boolean circuits D of size $|D| \leq s$ we have $|\Pr[D(X)=1] - \Pr[D(Y)=1]| \leq \epsilon$. The statistical distance of X and Y is $d_1(X;Y) \stackrel{\mathsf{def}}{=} \sum_x |P_X(x) - P_Y(x)|$ (where $P_X(x) \stackrel{\mathsf{def}}{=} \Pr[X=x]$), the Euclidean distance of X and Y is $d_2(P_X;P_Y) \stackrel{\mathsf{def}}{=} \sqrt{\sum_x (P_X(x)-P_Y(x))^2}$. A variable X has min-entropy k if it doesn't take any particular outcome with probability greater 2^{-k}, it has δ-smooth min-entropy k [6], if it's δ close to some distribution with min-entropy k. X has k bits of HILL pseudoentoentry of quality (s,ϵ) if there exists a Y with min-entropy k that is (s,ϵ) indistinguishable from X, we use the following standard notation for these notions:

min-entropy: $\mathsf{H}_\infty(X) \stackrel{\mathsf{def}}{=} -\log \max_x (\Pr[X=x])$.

smooth min-entropy: $\mathsf{H}^\delta_\infty(X) \stackrel{\mathsf{def}}{=} \max_{Y, d_1(X;Y)\leq\delta} \mathsf{H}_\infty(Y)$.

HILL pseudoentropy: $\mathsf{H}^{\mathsf{HILL}}_{s,\epsilon}(X) \stackrel{\mathsf{def}}{=} \max_{Y, Y\sim_{(s,\epsilon)}X} \mathsf{H}_\infty(Y)$.

1.2 Our Contribution

In this work give generic non-uniform attacks on pseudoentropy distributions. A seemingly natural goal is to consider a distribution X with $\mathsf{H}_\infty(X) \leq k$ bits of min-entropy, strictly larger $\mathsf{H}^{\mathsf{HILL}}_{s,\epsilon}(X) \geq k+1$ bits of HILL entropy, and then give an upper bound on s in terms of ϵ. This does not work as there are X where $\mathsf{H}_\infty(X) \ll \mathsf{H}^\delta_\infty(X)$,[1] and as by definition $\mathsf{H}^\delta_\infty(X) = \mathsf{H}^{\mathsf{HILL}}_{\infty,\delta}(X)$, we can have a large entropy gap $\mathsf{H}^{\mathsf{HILL}}_{\infty,\delta}(X) - \mathsf{H}_\infty(X)$ even when considering unbounded adversaries against HILL entropy. For this reason, in our main technical result 1 below, we must consider distributions with bounded *smooth* min-entropy. This makes the statement of the lemma somewhat technical. In practice, the distributions considered often have bounded support, for example because they were generated from a short seed by a deterministic process (like a pseudorandom generator). In this case we can drop the smoothness requirement as stated in Theorem 2 below.

▶ **Lemma 1** (Nonuniform attacks against pseudoentropy). *Suppose that $X \in \{0,1\}^n$ does not have k bits of δ-smooth min-entropy, i.e., $\mathsf{H}^\delta_\infty(X) < k$, then for any ϵ we have*

$$\mathsf{H}^{\mathsf{HILL}}_{\tilde{O}(2^k\epsilon^2\delta^{-2}),\epsilon}(X) < k$$

where $\tilde{O}(\cdot)$ hides a factor linear in n.

▶ **Theorem 2.** *Let $f : \{0,1\}^k \to \{0,1\}^n$ be a deterministic (not necessarily efficient) function. Then we have*

$$\mathsf{H}^{\mathsf{HILL}}_{\tilde{O}(2^k\epsilon^2),\epsilon}(f(U_k)) \leq k+1.$$

more generally, for any X over $\{0,1\}^n$ with support of size $\leq 2^k$

$$\mathsf{H}^{\mathsf{HILL}}_{\tilde{O}(2^k\epsilon^2),\epsilon}(X) \leq k+1.$$

[1] Consider an X which is basically uniform over $\{0,1\}^n$, but has mass δ on one particular point, then $\log \delta^{-1} = \mathsf{H}_\infty(X) \ll \mathsf{H}^\delta_\infty(X) = n$.

▶ Remark (Concluding best attacks against PRGs). For the special case $n = k + 1$ we recover the bound for pseudo*random* generators from [2].

Proof of Theorem 2. The theorem follows from Lemma 1 when $\delta = 1/2$; consider any X with support of size $\leq 2^k$, then $\mathsf{H}^{\delta}_{\infty}(X) \leq k + 1$, as no matter how we cut probability mass of $1 - \delta = 1/2$ over 2^k elements, one element will have the weight at least 2^{-k-1}. ◀

1.3 Proof Outline

1.3.1 A Weaker Result as a Ball-Bins Problem

We outline the proof of a somewhat weakened version of Theorem 2 in the language of balls and bins. For every Y of min-entropy $k' = k + \Omega(1)$ we want to distinguish Y from $X = f(U_k)$. Suppose for simplicity that Y is flat and f is injective, so that X is also flat. Our strategy will be to hash the points randomly into two bins and take advantage of the fact that the *average maximum load* is closer to $\frac{1}{2}$ when we sample from Y than when drawing from X. The reason is that Y has more balls, so by the law of large numbers, we expect the load to be "more concentrated" around the mean.

Think of throwing balls (inputs x) into two bins (labeled by -1 and 1). If the balls come from the support of X, the expected maximum load (over two bins) equals $\approx 2^{k-1} + \sqrt{2/\pi} \cdot 2^{k/2}$. Similarly, if the balls come from the support of Y, then maximum load is $2^{k'-1} + \sqrt{2/\pi} \cdot 2^{k'/2}$. In terms of the average load (the load normalized by the total number of balls):

$$\mathsf{AverageMaxLoad}(X) \approx 0.5 + \sqrt{2/\pi} \cdot 2^{-k/2} \quad \text{w.h.p. when drawing from } X\,,$$
$$\mathsf{AverageMaxLoad}(Y) \approx 0.5 + \sqrt{2/\pi} \cdot 2^{-k'/2} \quad \text{w.h.p. when drawing from } Y\,.$$

As $k' = k + \Omega(1)$ we obtain (with good probability):

$$\mathsf{AverageMaxLoad}(X) - \mathsf{AverageMaxLoad}(Y) = \Omega(2^{-k/2}).$$

Letting D be one of these bins assignments we obtain a distinguisher with advantage $\epsilon = \Omega(2^{-k/2})$. To generate the assignments efficiently we relax the assumption about choosing bins and assume only that the choices of bins are independent for any group of $\ell = 4$ balls. The fourth moment method allows us to keep sufficiently good probabilistic guarantees on the maximum load.

1.3.2 The General Case by Random Walk Techniques

1.3.2.1 A high-level outline and comparison to [2]

Below in Figure 1 we sketch the flow of our argument.

Our starting point is the proof from [2]. They use the fact that a random mapping $\mathsf{D} : \{0,1\}^n \to \{-1,1\}$ likely distinguishes any two distributions X and Y over $\{0,1\}^n$ with advantage being the Euclidean distance $d_2(X;Y) \stackrel{\mathsf{def}}{=} \sqrt{\sum_x (P_X(x) - P_Y(x))^2}$.

For any X and Y with constant statistical distance $\sum_x |P_X(x) - P_Y(x)| = \Theta(1)$ (which is the case for the PRG setting where $Y = U_n$ and $X = PRG(U_{n-1})$) this yields a bound $\Omega\left(2^{-\frac{n}{2}}\right)$. This bound can be then amplified, at the cost of extra advice, by partitioning the domain $\{0,1\}^n$ and combining corresponding advantages (advice basically encodes if there is a need for flipping the output). Finally one can show that 4-wise independence provides enough randomness for this argument, which makes sampling D efficient. Our argument deviates from this approach in two important aspects.

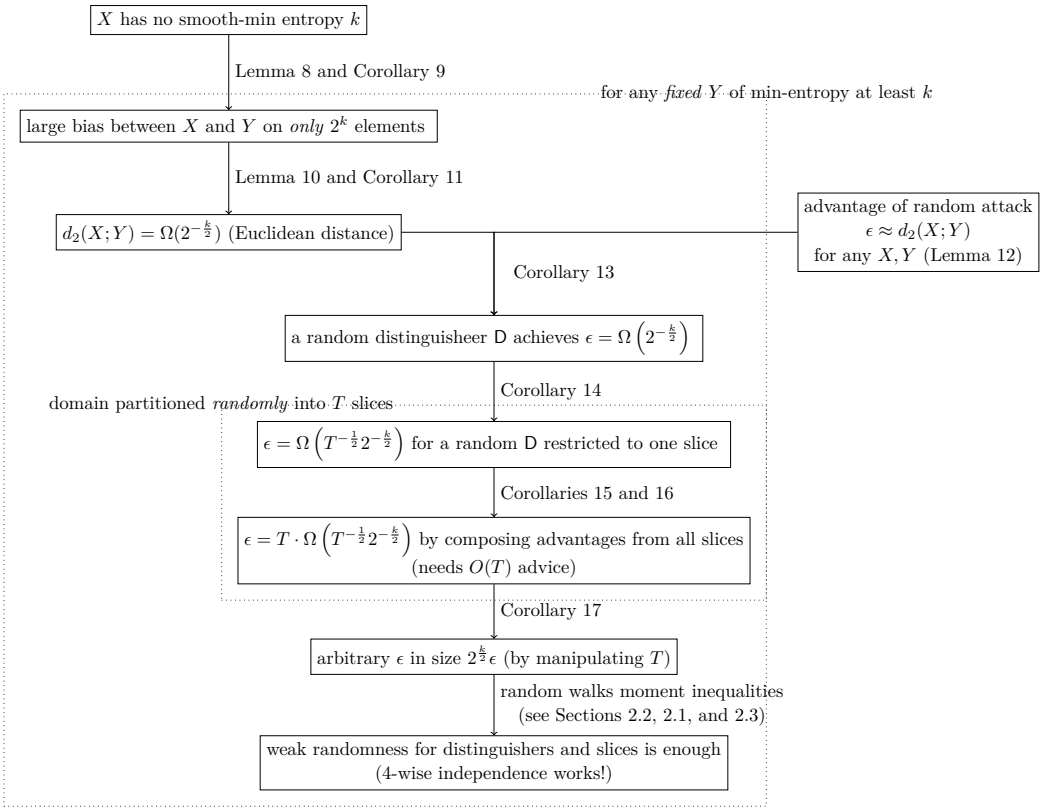


Figure 1 The map of our proof.

The first difference is that in the pseudoentropy case we can improve the advantage from $\Omega\left(2^{-\frac{n}{2}}\right)$, where n is the logarithm of the support of the variables considered, to $\Omega\left(2^{-\frac{k}{2}}\right)$, where k is the min-entropy of the variable we want to distinguish from. The reason is that being statistically far from any k-bit min-entropy distributions implies a *large bias on already* 2^k *elements*. This fact (see Lemma 8 and Corollary 9, and also Figure 3) is a new characterization of smooth min-entropy of independent interest.

The second subtlety arises when it comes to amplify the advantage over the partition slices. For the pseudorandomness case it is enough to split the domain in a deterministic way, for example by fixing prefixes of n-bit strings, in our case this is not sufficient. For us a "good" partition must shatter the 2^k-element high-biased set, which can be arbitrary. Our solution is to use *random partitions*, in fact, we show that using 4-universal hashing is sufficient. Generating base distinguishers and partitions at the same time makes probability calculations more involved.

Technical calculations are based on the fourth moment method, similarly as in [2]. The basic idea is that for settings where the second and fourth moment are easy to compute (e.g. sums of independent symmetric random variables) we can obtain good upper and lower bounds on the first moment. In the context of algorithmic applications these techniques are usually credited to [1]. Interestingly, exploiting natural relations to *random walks*, we show that calculations immediately follow by adopting classical (almost one century old) tools and results [5, 4]. Our technical novelty is an application of moment inequalities due to Marcinkiewicz-Zygmund and Paley-Zygmund, which allow us to prove slightly more than just the existence of an attack. Namely we generate it with constant success probability.

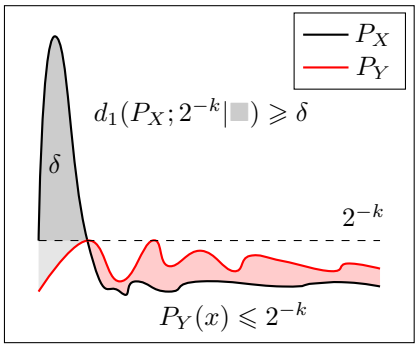


Figure 2 An intuition behind the attack. Random ± 1-weights make the bias equal to the ℓ_2-distance of P_X and P_Y. This distance can be bounded in terms of the ℓ_1 distance, which concentrates mass difference δ on less than 2^k elements (the region in gray).

1.3.2.2 Advantage $\Omega(2^{-k/2})$

Consider any X with δ-smooth min-entropy smaller than k. This requirement can be seen as a statement about the "shape" of the distribution. Namely, the mass of X that is above the threshold 2^{-k} equals at least δ, that is

$$\sum_x \max(P_X(x) - 2^{-k}, 0) \geqslant \delta.$$

For an illustration see Figure 2.

We construct our attack based on this observation. Define the advantage of a function D for distributions X and Y as

$$\mathsf{Adv}^{\mathsf{D}}(X;Y) = \left| \sum_x \mathsf{D}(x)(P_X(x) - P_Y(x)) \right|$$

(writing also $\mathsf{Adv}^{\mathsf{D}}_S$ when the summation is restricted to a subset S). Consider a random distinguisher $\mathsf{D} : \{0,1\}^n \to \{-1,1\}$. Random variables $\mathsf{D}(x)$ for different x are independent, have zero-mean and second moment equal to 1. Therefore the expected square of of the advantage, over the choice of D, equals

$$\mathbb{E}\left[\left(\mathsf{Adv}^{\mathsf{D}}(X;Y)\right)^2\right] = \mathbb{E}\left|\sum_x \mathsf{D}(x)(P_X(x) - P_Y(x))\right|^2 = \sum_x (P_X(x) - P_Y(x))^2 .$$

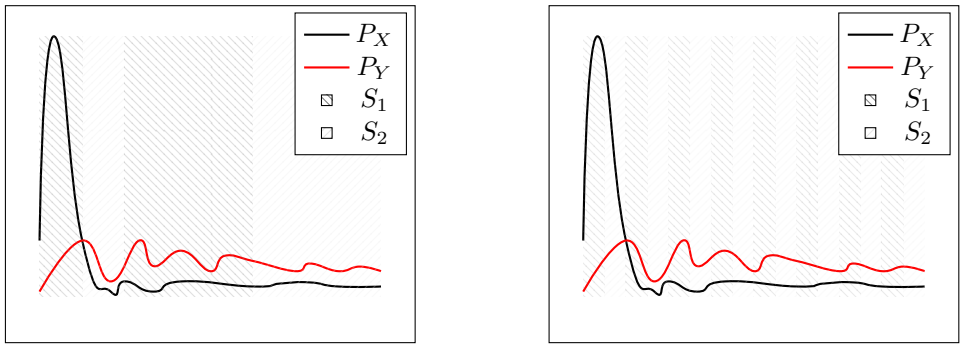
Let S be the set of x such that $P_X(x) > 2^{-k}$. For any Y of min-entropy at least k we obtain

$$\sum_{x\in S}(P_X(x) - P_Y(x))^2 \geqslant \sum_{x\in S}(P_X(x) - 2^{-k})^2 \geqslant |S|^{-1}\left(\sum_{x\in S}\left(P_X(x) - 2^{-k}\right)\right)^2 \geqslant 2^{-k}\delta^2$$

where the first inequality follows because $P_Y(x) \leqslant 2^{-k} < P_X(x)$ for $x \in S$, the second inequality is by the standard inequality between the first and second norm, and the third inequality follows because we showed that $\Pr[X \in S] \geqslant |S| \cdot 2^{-k} + \delta$ (illustrated in Figure 2) which also implies $|S|^{-1} \geqslant 2^{-k}$.

By the previous formula on the expected squared advantage this means that

$$\mathbb{E}\left[\left(\mathsf{Adv}^{\mathsf{D}}(X;Y)\right)^2\right] \geqslant 2^{-k}\delta^2$$



(a) An example of a "bad" partition. Almost all advantage is captured by one partition slice S_1.

(b) An example of a "good" partition. The advantage is evenly distributed among slices S_1, S_2.

Figure 3 Illustration of good and bad partitions.

for at least one choice of D. This implies

$$\mathsf{Adv}^{\mathsf{D}}(X;Y) \geqslant 2^{-\frac{k}{2}}\delta.$$

A random D as defined would be of size exponential in n, but since we used only the second moment in calculations, it suffices to generate $\mathsf{D}(x)$ as pairwise independent random variables. By assuming 4-wise independence – which can be computed by $O(n^2)$ size circuits – we can prove slightly more, namely that a constant fraction of generated D's are good distinguishers. This property will be important for the next step, where we amplify the advantage assuming larger distinguishers.

1.3.2.3 Leveraging the advantage by slicing the domain

Consider a random and equitable partition $\{S_i\}_{i=1}^{T}$ of the set $\{0,1\}^n$. From the previous analysis we know that a random distinguisher achieves advantage $\epsilon = d_2(P_X;P_Y)$ over the whole domain. Note that (for any, not necessarily random partition $\{S_i\}_i$) we have

$$\left(d_2(P_X;P_Y)\right)^2 = \sum_{i=1}^{T} \left(d_2(P_X;P_Y|S_i)\right)^2$$

where $d_2(P_X;P_Y|S_i)$ is the restriction of the distance to the set S_i (by restricting the summation to S_i). From a random partition we expect the mass difference between P_X and P_Y to be *distributed evenly* among the partition slices (see Figure 3(b)). Based on the last equation, we expect

$$d_2(P_X;P_Y|S_i) \approx \frac{d_2(P_X;P_Y)}{\sqrt{T}}$$

to hold with high probability over $\{S_i\}_i$.

In fact, if the mass difference is not well balanced amongst the slices (in the extreme case, concentrated on one slice) our argument will not offer any gain over the previous construction (see Figure 3(a)).

By applying the previous argument to individual slices, for every i we can obtain an advantage $\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) = \Omega\left((T^{-\frac{1}{2}}2^{-\frac{k}{2}})\delta\right)$ when restricted to the set S_i (with high probability

over the choice of D and $\{S_i\}_i$). Now if the sets S_i are *efficiently recognizable*, we can combine them into a better distinguisher. Namely for every i we chose a value $\beta_i \in \{-1,1\}$ such that D's advantage (before taking the absolute value) restricted to S_i has sign β_i, and set

$$\hat{\mathsf{D}}(x) = \beta_i \mathsf{D}(x), \text{ where } i \text{ is such that } x \in S_i,$$

then the advantage equals (with high probability over D and the S_i's)

$$\mathsf{Adv}^{\hat{\mathsf{D}}}(X;Y) = \sum_{i=1}^{T} \mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) = \Omega\left(T^{\frac{1}{2}} 2^{-\frac{k}{2}} \delta\right).$$

We need to specify a 4-wise independent hash for D, another 4-wise independent hash for deciding in which of the T slices an element lies, and T bits to encode the β_i's. Thus for a given T the size of $\hat{\mathsf{D}}$ will be $T + \tilde{O}(n)$. Using the above equation, we then get a smooth tradeoff $s = O(2^k \epsilon^2 \delta^{-2})$ between the advantage ϵ and the circuit size s. This discussion shows that to complete the argument we need the following two properties of the partition (a) the mass difference between P_X and P_Y is (roughly) equidistributed among slices and (b) the membership in partition slices can be efficiently decided.

1.3.2.4 Slicing using 4-wise independence

To complete the argument, we assume that T is a power of 2, and generate the slicing by using a 4-universal hash function $h : \{0,1\}^n \to \{0,1\}^{\log T}$. The i-th slice S_i is defined as $\{x \in \{0,1\}^n : h(x) = i\}$. These assumptions are enough to prove that

$$\mathbb{E}\, \mathsf{Adv}^{\hat{\mathsf{D}}}_{S_i}(X;Y) = \Omega\left(T^{-\frac{1}{2}} d_2(P_X;P_Y)\right) = \Omega\left(T^{-\frac{1}{2}} 2^{-\frac{k}{2}} \delta\right).$$

Interestingly, the expected advantage (left-hand side) cannot be computed directly. The trick here is to bound it in terms of the second and fourth moment. The above inequality, coupled with bounds on second moments of the advantage $\mathsf{Adv}^{\hat{\mathsf{D}}}_{S_i}$ (obtained directly), allows us to prove that

$$\Pr\left[\sum_{i=1}^{T} \mathsf{Adv}^{\hat{\mathsf{D}}}_{S_i} \geqslant \Omega(1) \cdot T^{\frac{1}{2}} 2^{-\frac{k}{2}} \delta\right] > \Omega(1).$$

This shows that there exists the claimed distinguisher $\hat{\mathsf{D}}$. In fact, a *constant fraction* of generated (over the choice of D and $\{\S_i\}_i$) distinguishers $\hat{\mathsf{D}}$'s works.

1.3.2.5 Random walks

From a technical point of view, our method involves computing higher moments of the advantages to obtain concentration and anti-concentration results. The key observation is that the advantage written down as

$$\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) = \left|\sum_x (P_X(x) - P_Y(x)) \mathbf{1}_{S_i}(x) \mathsf{D}(x)\right|$$

which can be then studied as a *random walk*

$$\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) = \left|\sum_x \xi_{i,x}\right|$$

with zero-mean increments $\xi_{i,x} = (P_X(x) - P_Y(x)) \mathbf{1}_{S_i}(x) \mathsf{D}(x)$. The difference with respect to classical model is that the increments are only ℓ-wise independent (for $\ell = 4$). However, that classical moment bounds still apply (see Sections 2.2 and 2.3 for more details).

2 Preliminaries

2.1 Interpolation Inequalities

Interpolation inequalities show how to bound the p-th moment of a random variable if we know bounds on one smaller and one higher moment. The following result is known also as *log-convexity of L_p norms*, and can be proved by the Hölder Inequality.

▶ **Lemma 3** (Moments interpolation). *For any $p_1 < p < p_2$ and any bounded random variable Z we have*

$$\|Z\|_p \leqslant (\|Z\|_{p_1})^{\theta} (\|Z\|_{p_2})^{1-\theta}$$

where θ is such that $\frac{\theta}{p_1} + \frac{1-\theta}{p_2} = \frac{1}{p}$, and for any r we define $\|Z\|_r = (\mathbb{E}\,|Z|^r)^{\frac{1}{r}}$.

Alternatively, we can *lower bound* a moment given *two higher moments*. This is very useful when higher moments are easier to compute. In this work will bound first moments from below when we know the second and the fourth moment (which are easier to compute as they are even-order moments)

▶ **Corollary 4.** *For any bounded Z we have $\mathbb{E}\,|Z| \geqslant \frac{(\mathbb{E}\,|Z|^2)^{\frac{3}{2}}}{(\mathbb{E}\,|Z|^4)^{\frac{1}{2}}}$.*

2.2 Moments of random walks

For a random walk $\sum_x \xi(x)$, where $\xi(x)$ are independent with zero-mean, we have good control over the moments, namely $\mathbb{E}\,|\sum_x \xi(x)|^p = \Theta(1) \cdot (\sum_x \mathrm{Var}(\xi(x)))^{\frac{p}{2}}$ where constants depend on p. This result is due to Marcinkiewicz and Zygmund [5] who extended the former result of Khintchine [4]. Below we notice that for *small moments* p it suffices to assume only p-wise independence (most often used versions assume fully independence)

▶ **Lemma 5** (Strengthening of Marcinkiewicz-Zygmund's Inequality for $p = 4$). *Suppose that $\{\xi(x)\}_{x\in\mathcal{X}}$ are 4-wise independent, with zero mean. Then we have*

$$\frac{1}{\sqrt{3}} \left(\sum_{x\in\mathcal{X}} \mathrm{Var}(\xi(x))\right)^{\frac{1}{2}} \leqslant \mathbb{E}\left|\sum_{x\in\mathcal{X}} \xi(x)\right| \leqslant \left(\sum_{x\in\mathcal{X}} \mathrm{Var}(\xi(x))\right)^{\frac{1}{2}},$$

$$\mathbb{E}\left|\sum_{x\in\mathcal{X}} \xi(x)\right|^2 = \sum_{x\in\mathcal{X}} \mathrm{Var}(\xi(x)),$$

$$\left(\sum_{x\in\mathcal{X}} \mathrm{Var}(\xi(x))\right)^2 \leqslant \mathbb{E}\left|\sum_{x\in\mathcal{X}} \xi(x)\right|^4 \leqslant 3\left(\sum_{x\in\mathcal{X}} \mathrm{Var}(\xi(x))\right)^2.$$

The proof appears in Section 4.1.

2.3 Anticontentration bounds

▶ **Lemma 6** (Paley-Zygmund Inequality). *For any positive random variable Z and a parameter $\theta \in (0,1)$ we have*

$$\Pr\left[Z > \theta\,\mathbb{E}\,Z\right] \geqslant (1-\theta)^2 \frac{(\mathbb{E}\,Z)^2}{\mathbb{E}\,Z^2}.$$

By applying Lemma 6 to the setting of Lemma 5, and choosing $\theta = \frac{1}{\sqrt{3}}$ we obtain:

▶ **Corollary 7** (Anticoncentration for walks with 4-wise independent increments). *Suppose that* $\{\xi(x)\}_{x\in\mathcal{X}}$ *are* 4*-wise independent with zero-mean, then we have*

$$\Pr\left[\left|\sum \xi(x)\right| > \frac{1}{3}\left(\sum \mathrm{Var}(\xi(x))\right)^{\frac{1}{2}}\right] > \frac{1}{17}.$$

where the summation is over $x \in \mathcal{X}$.

3 Proof of Lemma 1

▶ **Lemma 8** (Characterizing smooth min-entropy). *For any random variable* X *with values in a finite set* $\mathcal{X}$*, any* δ *and* k *we have the following equivalence*

$$H_\infty^\delta(X) \geqslant k \iff \sum_{x\in\mathcal{X}} \max\left(P_X(x) - 2^{-k}, 0\right) \leqslant \delta.$$

The proof appears in Section 4.2. We will work with the following equivalent statement

▶ **Corollary 9** (No smooth min-entropy k implies bias w.r.t. distributions of min-entropy k over at most 2^k elements). *We have* $\mathsf{H}_\infty^\delta(X) < k$ *if and only if there exists a set* S *of at most* 2^k *elements such that*

$$\sum_{x\in S} |P_X(x) - P_Y(x)| > \delta$$

for all Y *of min-entropy at least* k.

Proof of Corollary 9. The direction $\Longleftarrow$ trivially follows by the definition of smooth min-entropy. Now assume $\mathsf{H}_\infty^\delta(X) < k$. Let S be the set of all x such that $P_X(x) > 2^{-k}$, then $|S| < 2^k$, and moreover by Lemma 8 we have $\sum_{x\in S}\left(P_X(x) - 2^{-k}\right) > \delta$. In particular for any Y of min-entropy k (i.e., $P_Y(x) \leqslant 2^{-k}$ for all x)

$$\sum_{x\in S} (P_X(x) - P_Y(x)) > \delta\,. \qquad \blacktriangleleft$$

▶ **Lemma 10** (Bias implies Euclidean distance). *For any distributions* P_X, P_Y *on* $\mathcal{X}$ *and any subset* S *of* $\mathcal{X}$ *we have*

$$\left(\sum_{x\in S} (P_X(x) - P_Y(x))^2\right)^{\frac{1}{2}} > |S|^{-1/2} \sum_{x\in S} |P_X(x) - P_Y(x)|\,.$$

Proof. By the Jensen Inequality we have

$$|S|^{-1}\left(\sum_{x\in S} (P_X(x) - P_Y(x))^2\right) > \left(|S|^{-1}\sum_{x\in S} |P_X(x) - P_Y(x)|\right)^2$$

which is equivalent to the statement. ◀

▶ **Corollary 11** (No smooth min-entropy implies Euclidean distance to min-entropy distributions). *Suppose that* $\mathsf{H}_\infty^\delta(X) < k$*. Then for any* Y *of min-entropy at least* k *we have* $\left(\sum_x |P_X(x) - P_Y(x)|^2\right)^{\frac{1}{2}} > 2^{-\frac{k}{2}}\delta$.

Proof of Corollary 11. It suffices to combine Lemma 10 and Corollary 9. ◀

By Corollary 7 we conclude that the advantage of a random distinguisher for any two measures (in our case P_X and P_Y) equals the Euclidean distance.

▶ **Lemma 12** (The advantage of a random distinguisher equals the Euclidean distance). *Let $\{\mathsf{D}(x)\}_{x\in\{0,1\}^n}$ be 4-wise independent as indexed by x and such that $\mathsf{D}(x)$ outputs a random element from $\{-1,1\}$. Then for any set S we have*

$$\left|\sum_{x\in S}\mathsf{D}(x)(P_X(x)-P_Y(x))\right| > \frac{1}{3}\cdot d_2(P_X;P_Y)$$

with probability $\frac{1}{17}$ over the choice of D (the result actually holds for any measures in place of P_X, P_Y).

For our case, that is the setting in Lemma 10, we obtain

▶ **Corollary 13** (A random attack achieves $\Omega\left(2^{-k}\delta\right)$ with significant probability). *For X,Y as in Corollary 11, and D as in Lemma 12 we have $\mathsf{Adv}^{\mathsf{D}}(X;Y) \geq \frac{1}{3}\cdot 2^{-\frac{k}{2}}\delta$ w.p. $\frac{1}{17}$ over D.*

3.1 Partitioning the domain into T slices

Let $h : \{0,1\}^n \to [1\ldots 2^t]$, where $t = \lceil \log T \rceil$, be a 4-universal hash function. Define $S_i = \{x : h(x) = i\}$, $\Delta(x) = P_X(x) - P_Y(x)$ and consider advantages on slices S_i

$$\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) = \left|\sum_x \Delta(x)\mathsf{D}(x)\mathbf{1}_{S_i}(x)\right| .$$

The following corollary shows that on each of our T slices, we get the advantage $T^{-\frac{1}{2}}2^{-\frac{k}{2}}\delta$. The proof appears in Section 4.3.

▶ **Corollary 14** ((Mixed) moments of slice advantages). *For D, $\{S_u\}_u$ as above and every i,j*

$$\mathbb{E}_{\mathsf{D},\{S_u\}_u}\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) \geqslant 3^{-\frac{1}{2}}T^{-\frac{1}{2}}\cdot d_2\left(P_X;P_Y\right),$$
$$\mathbb{E}_{\mathsf{D},\{S_u\}_u}\left(\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y)\,\mathsf{Adv}^{\mathsf{D}}_{S_j}(X;Y)\right) \leqslant T^{-1}\cdot d_2(P_X;P_Y)^2,$$

(the statement is valid for arbitrary measures in place of P_X, P_Y).

Denote $Z = \sum_i \mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y)$. Using Lemma 6 with $\theta = \frac{1}{\sqrt{3}}$ where we compute $\mathbb{E}\,Z^2$ and $\mathbb{E}\,Z$ according to Corollary 14 we obtain $\Pr\left[|Z| > \frac{1}{\sqrt{3}}\cdot\mathbb{E}\,|Z|\right] \geqslant \frac{1}{17}$. Bounding once again $\mathbb{E}\,|Z|$ as in Corollary 14 we get

▶ **Corollary 15** (Total advantage on all partition slices). *For X,Y as in Corollary 11, D and S_i defined above we have*

$$\Pr_{\mathsf{D},\{S_u\}_u}\left[\sum_{i=1}^{T}\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) \geqslant \frac{1}{3}\cdot T^{\frac{1}{2}}2^{-\frac{k}{2}}\delta\right] \geqslant \frac{1}{17}$$

(for general X,Y the lower bound is $\Omega(1)\cdot T^{\frac{1}{2}}\cdot d_2(P_X;P_Y)$).

The corollary shows that the *total absolute advantage* over all partition slices, is as expected. Since $\{S_i\}_i$ is a partition we have

$$\sum_{i=1}^{T}\mathsf{Adv}^{\mathsf{D}}_{S_i}(X;Y) = \sum_{i=1}^{T}\left|\sum_{x\in S_i}\left(P_X(x)-P_Y(x)\right)\mathsf{D}(x)\right| = \sum_x\left(P_X(x)-P_Y(x)\right)\mathsf{D}(x)\beta(x)$$

where for $\beta_i \stackrel{def}{=} \text{sgn}\left(\sum_{x\in S_i}(P_X(x')-P_Y(x))\,\mathsf{D}(x)\right)$ (the sign of the advantage on the i-th slice) we define $\beta(x)=\beta_i$ where S_i contains x. This shows that by "flipping" the distinguisher output on the slices we achieve the sum of individual advantages. Since the bit $\beta(x)$ can be computed with $O(T)+\tilde{O}(n)$ advice (the complexity of the function $i\to\beta_i$ plus the complexity of finding i for a given x) we obtain

▶ **Corollary 16** (Computing total advantage by one distinguisher). *For X,Y as in Corollary 11, D and $\{S_i\}_i$ defined above there exists a modification to D which in time $\tilde{O}(n)$ and advice $O(T)$ achieves advantage $\frac{1}{3}\cdot T^{\frac{1}{2}}2^{-\frac{k}{2}}\delta$ with probability $\frac{1}{17}$.*

Finally by setting $\epsilon = T^{\frac{1}{2}}2^{-\frac{k}{2}}\delta$ and manipulating T we arrive at

▶ **Corollary 17** (Continue tradeoff). *For any ϵ there exists T such that the distinguisher in Corollary 16 has advantage ϵ and circuit complexity $s=O\left(2^k\epsilon^2\delta^{-2}\right)$.*

4 Omitted Proofs

4.1 Proof of Lemma 5 (Strengthening of Marcinkiewicz-Zygmund's Inequality for $p=4$)

Let $Z=\sum_x \xi(x)$. Since $\xi(x)$ are (in particular) 2-wise independent with zero mean, we get

$$\mathbb{E}\left(\sum_x \xi(x)\right)^2 = \sum_{x,y}\mathbb{E}\left(\xi(x)\xi(y)\right) = \sum_{x=y}\mathbb{E}\left(\xi(x)\xi(y)\right) = \sum_x \text{Var}(\xi(x))\,,$$

(the summation taken over $x,y\in\mathcal{X}$). The fourth moment is somewhat more complicated

$$\begin{aligned}
\mathbb{E}\left(\sum_x \xi(x)\right)^4 &= \sum_{x_1,x_2,x_3,x_4}\mathbb{E}\left(\xi(x_1)\xi(x_2)\xi(x_3)\xi(x_4)\right)\\
&= \sum_{x_1=x_2=x_3=x_4}\mathbb{E}\left(\xi(x_1)\xi(x_2)\xi(x_3)\xi(x_4)\right)+\\
&\quad +3\sum_{x_1=x_2\neq x_3=x_4}\mathbb{E}\left(\xi(x_1)\xi(x_2)\xi(x_3)\xi(x_4)\right)\\
&= \sum_x \mathbb{E}\,\xi(x)^4 + 3\sum_{x\neq y}\mathbb{E}\,\xi(x)^2\,\mathbb{E}\,\xi(y)^2\\
&= 3\left(\sum_x \mathbb{E}\,\xi(x)^2\right)^2 - 2\sum_x \mathbb{E}\,\xi(x)^4\,.
\end{aligned}$$

The second equality follows because whenever $\xi(x)$ occurs in an odd power, for example $x=x_1\neq x_2=x_3=x_4$, the expectation is zero (this way one can simplify and bound also higher moments, see [7]). It remains to estimate the first moment. By Corollary 4 and bounds on the second and fourth moment we have just computed we obtain

$$\frac{1}{\sqrt{3}}\cdot\left(\sum_{x\in\mathcal{X}}\text{Var}(\xi(x))\right)^{\frac{1}{2}} \leqslant \mathbb{E}\left|\sum_{x\in\mathcal{X}}\xi(x)\right|$$

and the upper bound follows by Jensen's Inequality (with constant 1).

4.2 Proof of Lemma 8 (Characterizing smooth min-entropy)

Suppose that $\mathsf{H}_\infty^\delta(X) \geqslant k$. then, by definition, there is Y such that $\mathsf{H}_\infty(Y) \geqslant k$ and $\sum_{x:P_X(x)>P_Y(x)} P_X(x) - P_Y(x) \leqslant \delta$. Since all the summands are positive and since $P_Y(x) \leqslant 2^{-k}$, ignoring those x for which $P_Y(x) < 2^{-k}$ yields

$$\sum_{x:P_X(x)>2^{-k}} P_X(x) - P_Y(x) \leqslant \delta.$$

Again, since $P_Y(x) \leqslant 2^{-k}$ we obtain

$$\sum_{x:P_X(x)>2^{-k}} P_X(x) - 2^{-k} \leqslant \delta,$$

which finishes the proof of the "$\Longrightarrow$" part.

Assume now that $\delta' = \sum_{x\in\mathcal{X}} \max\left(P_X(x) - 2^{-k}, 0\right) \leqslant \delta$. Note that

$$\sum_{x\in\mathcal{X}} \max\left(P_X(x) - \frac{1}{2^k}, 0\right) + \sum_{x\in\mathcal{X}} \max\left(\frac{1}{2^k} - P_X(x), 0\right) = \\ = 2\sum_{x\in\mathcal{X}} \left|P_X(x) - \frac{1}{2^k}\right| \geqslant 2\sum_{x\in\mathcal{X}} \max\left(P_X(x) - \frac{1}{2^k}, 0\right)$$

and therefore we have $\sum_{x\in\mathcal{X}} \max\left(2^{-k} - P_X(x), 0\right) \geqslant \delta'$. By this observation we can construct a distribution Y by shifting δ' of the mass of P_X from the set $S^- = \{x : P_X(x) > 2^{-k}\}$ to the set $\{x : 2^{-k} \geqslant P_X(x)\}$ in such a way that we have $P_Y(x) \leqslant 2^{-k}$ for all x. Thus $\mathsf{H}_\infty(Y) \geqslant k$ and since a δ' fraction of the mass is shifted and redistributed we have $d_1(X;Y) \leqslant \delta'$. This finishes the proof of the "$\Longleftarrow$" part.

4.3 Proof of Corollary 14 ((Mixed) moments of slice advantages)

For shortness denote $\Delta(x) = P_X(x) - P_Y(x)$ and $\mathsf{Adv}_{S_i}^{\mathsf{D}} = \mathsf{Adv}_{S_i}^{\mathsf{D}}(X;Y)$.

Note that by Lemma 5, applied to the family $f_x = \Delta(x)\mathsf{D}(x)\mathbf{1}_{S_i}(x)$ (which is 4-wise independent) we have

$$\mathbb{E}\,\mathsf{Adv}_{S_i}^{\mathsf{D}} \geqslant 3^{-\frac{1}{2}} \left(\sum_x \Delta(x)^2\right)^{\frac{1}{2}}$$

which is the first inequality claimed in the corollary. In turn, again by Lemma 5, we have

$$\mathbb{E}\left(\mathsf{Adv}_{S_i}^{\mathsf{D}}\right)^2 = T^{-1} \cdot \sum_x \Delta(x)^2.$$

Since this holds for any i, by Cauchy-Schwarz we get for any i, j

$$\mathbb{E}\,\mathsf{Adv}_{S_i}^{\mathsf{D}}\mathsf{Adv}_{S_j}^{\mathsf{D}} \leqslant \sqrt{\mathbb{E}\left(\mathsf{Adv}_{S_i}^{\mathsf{D}}\right)^2 \cdot \mathbb{E}\left(\mathsf{Adv}_{S_j}^{\mathsf{D}}\right)^2} \leqslant T^{-1} \cdot \sum_x \Delta(x)^2$$

which proves the second inequality in the corollary.

References

1 Bonnie Berger. The fourth moment method. *SIAM J. Comput.*, 26(4):1188–1207, 1997. doi:10.1137/S0097539792240005.

2 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time Space Tradeoffs for Attacks against One-Way Functions and PRGs. In *Advances in Cryptology – CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 649–665, 2010. doi:10.1007/978-3-642-14623-7_35.

3 Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. Key derivation without entropy waste. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 93–110. Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-642-55220-5_6.

4 Aleksandr Khintchine. Über einen Satz der Wahrscheinlichkeitsrechnung. *Fundamenta Mathematicae*, 6(1):9–20, 1924. URL: http://eudml.org/doc/214283.

5 J. Marcinkiewicz and A. Zygmund. Quelques théorèmes sur les fonctions indépendantes. *Studia Mathematica*, 7(1):104–120, 1938. URL: http://eudml.org/doc/218615.

6 Renato Renner and Stefan Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In *Advances in Cryptology – ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, pages 199–216, 2005. doi:10.1007/11593447_11.

7 Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas.*, pages 331–340, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313797.

Interactive Oracle Proofs with Constant Rate and Query Complexity*†

Eli Ben-Sasson[1], Alessandro Chiesa[2], Ariel Gabizon[‡3], Michael Riabzev[4], and Nicholas Spooner[5]

1 **Technion, Haifa, Israel**
eli@cs.technion.ac.il
2 **University of California, Berkeley, CA, USA**
alexch@berkeley.edu
3 **Zerocoin Electronic Coin Company (Zcash), Boulder, CO, USA**
ariel@z.cash
4 **Technion, Haifa, Israel**
mriabzev@cs.technion.ac.il
5 **University of Toronto, Toronto, Canada**
spooner@cs.toronto.edu

Abstract

We study *interactive oracle proofs* (IOPs) [7, 43], which combine aspects of probabilistically checkable proofs (PCPs) and interactive proofs (IPs). We present IOP constructions and techniques that let us achieve tradeoffs in proof length versus query complexity that are not known to be achievable via PCPs or IPs alone. Our main results are:

1. *Circuit satisfiability has* 3*-round IOPs with linear proof length (counted in bits) and constant query complexity.*
2. *Reed–Solomon codes have 2-round IOPs of proximity with linear proof length and constant query complexity.*
3. *Tensor product codes have 1-round IOPs of proximity with sublinear proof length and constant query complexity.* (A familiar example of a tensor product code is the Reed–Muller code with a bound on individual degrees.)

For all the above, known PCP constructions give *quasilinear* proof length and constant query complexity [12, 16]. Also, for circuit satisfiability, [10] obtain PCPs with linear proof length but *sublinear* (and super-constant) query complexity. As in [10], we rely on algebraic-geometry codes to obtain our first result; but, unlike that work, our use of such codes is much "lighter" because we do not rely on any automorphisms of the code.

We obtain our results by building "IOP-analogues" of tools underlying numerous IPs and PCPs:

- **Interactive proof composition.** Proof composition [3] is used to reduce the query complexity of PCP verifiers, at the cost of increasing proof length by an additive factor that is exponential in the verifier's randomness complexity. We prove a composition theorem for IOPs where this additive factor is linear.
- **Sublinear sumcheck.** The sumcheck protocol [34, 46] is an IP that enables the verifier to check the sum of values of a low-degree multi-variate polynomial on an exponentially-large hypercube, but the verifier's running time depends linearly on the bound on individual degrees. We prove a sumcheck protocol for IOPs where this dependence is sublinear (e.g., polylogarithmic).

* A full version of the paper is available at https://eprint.iacr.org/2016/324.

† This work has received funding from the Israel Science Foundation (grant 1501/14), the UC Berkeley Center for Long-Term Cybersecurity, and the United States – Israel Binational Science Foundation (grant 2021036).

‡ This work was done while Ariel Gabizon was at Technion and visiting UC Berkeley.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 40; pp. 40:1–40:15

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

Our work demonstrates that even constant-round IOPs are more efficient than known PCPs and IPs.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases probabilistically checkable proofs, interactive proofs, proof composition, sumcheck

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.40

1 Introduction

We study *Interactive Oracle Proofs* (also known as *Probabilistically Checkable Interactive Proofs*) [7, 43], which combine aspects of probabilistically checkable proofs (PCPs) and interactive proofs (IPs). We present IOP constructions and general techniques that enable us to obtain tradeoffs in proof length versus query complexity that are not known to be achievable by either PCPs or IPs alone. For some applications (e.g., constructing non-interactive arguments in the random oracle model [7]) considering such general types of proof systems suffices (as opposed to focusing only on PCPs or IPs) and thus these applications inherit the efficiency improvements over PCPs.

1.1 Motivation

Probabilistically checkable proofs (PCPs) were introduced by [22, 5, 20, 3, 2]: in a PCP, a probabilistic polynomial-time verifier has oracle access to the proof string. The complexity class $\mathbf{PCP}[\mathsf{r}, \mathsf{q}]$ denotes those languages for which the verifier uses at most r random bits and queries at most q proof locations; the proof length is then at most 2^{r}. The PCP Theorem [3, 2] states that $\mathbf{NP} = \mathbf{PCP}[O(\log n), O(1)]$: every NP statement has a proof of polynomial length that can be verified via a constant number of queries (say, with soundness error $1/2$).

A fundamental question is how long a PCP needs to be, compared to the corresponding "standard" NP proof. Given $T\colon \mathbb{N} \to \mathbb{N}$, the PCP Theorem states that every language $\mathcal{L}$ in $\mathbf{NTIME}(T)$ has a proof of length $\mathrm{poly}(T(n))$ that can be verified with $O(1)$ queries. A sequence of works [42, 30, 24, 13, 9, 12, 16] gradually reduced the proof length to quasilinear, i.e., $T(n) \cdot \mathrm{polylog}(T(n))$; much of this progress was accompanied by progress on efficient reductions from **NTIME** to "PCP-friendly" problems, as well as efficient constructions of PCPs of proximity (PCPPs) for key classes of linear codes. Despite much progress, the following question remains open: *are there PCPs with linear proof length and constant query complexity?*

Ben-Sasson et al. [10] make progress in this direction by proving that there is $a > 0$ such that for every $\epsilon > 0$ there is a PCP for circuit satisfiability with proof length $2^{a/\epsilon} n$ and query complexity n^{ϵ}. Beyond the sublinear query complexity, [10]'s result comes with other caveats not affecting most prior constructions: the verifier is *non-uniform*, namely it requires a polynomial-size advice string for every circuit size; and the verifier is not succinct, namely it cannot run in time that is sublinear in the circuit size even if the circuit comes from a uniform circuit family. (Recent constructions of high-rate locally testable codes with sub-polynomial query complexity [32] are not yet known to be convertible to PCPs with similar parameters.)

In this paper, we continue the study of the tradeoff between proof length and query complexity, but we do so for a natural extension of the PCP model (sufficient for some useful

applications, e.g., [7]) that can be thought of as a "multi-round PCP", described below. Also, from this point onwards, we switch to using relations instead of languages. We denote by $\mathcal{R}$ a relation consisting of pairs $(\mathtt{x}, \mathtt{w})$, where $\mathtt{x}$ is the *instance* and $\mathtt{w}$ is the *witness*; we think of $\mathcal{R}$ naturally induced by a non-deterministic language $\mathcal{L}$. We denote by $\mathcal{R}|_{\mathtt{x}}$ the (possibly empty) set of witnesses for a given instance $\mathtt{x}$, and by n the size of $\mathtt{x}$.

1.2 A more general model: interactive oracle proofs

Interactive Oracle Proofs (IOPs) are a type of proof system introduced in [7, 43] that combines aspects of IPs [4, 26] and PCPs [5, 3, 2], and generalizes interactive PCPs [31]. IOPs naturally extend the notion of a PCP to multiple rounds or, viewed from an another angle, they naturally extend the notion of an IP by allowing probabilistic checking. Prior work shows that IOPs can be used to construct non-interactive proofs in the random oracle model [7], that IOPs efficiently achieve unconditional zero knowledge [6], and that IOPs can be used to obtain doubly-efficient constant-round IPs for polynomial-time bounded-space computations [43].

Informally, an IOP extends an IP as follows: whenever the prover sends to the verifier a message, the verifier does not have to read the message in full but may probabilistically query it. In more detail, a k-round IOP comprises k rounds of interaction. In the i-th round of interaction: the verifier sends a message m_i to the prover; then the prover replies with a message f_i to the verifier, which the verifier can query in this and all later rounds (by having oracle access to it). After the k rounds of interaction, the verifier either accepts or rejects.

An *IOP system* for a relation $\mathcal{R}$ with round complexity k and soundness ε is a pair (P, V), where P, V are probabilistic algorithms, that satisfies natural notions of completeness and soundness: for every instance-witness pair $(\mathtt{x}, \mathtt{w})$ in $\mathcal{R}$, $V(\mathtt{x})$ always accepts after $\mathsf{k}(n)$ rounds of interaction with $P(\mathtt{x}, \mathtt{w})$; and, for every instance $\mathtt{x}$ with $\mathcal{R}|_{\mathtt{x}} = \emptyset$ and unbounded prover $\tilde{P}$, $V(\mathtt{x})$ accepts with probability at most $\varepsilon(n)$ after $\mathsf{k}(n)$ rounds of interaction with $\tilde{P}$.

Like the IP model, one efficiency measure is the round complexity k. Like the PCP model, two additional efficiency measures are the *proof length* l, which is the total number of alphabet symbols in all of the prover's messages, and the *query complexity* q, which is the total number of locations queried by the verifier across all of the prover's messages. Considering all of these parameters, we say that a relation $\mathcal{R}$ belongs to the complexity class $\mathbf{IOP}[\mathsf{k}, \mathsf{a}, \mathsf{l}, \mathsf{r}, \mathsf{q}, \varepsilon]$ if there is an IOP system for $\mathcal{R}$ in which on instances of size n:

1. the number of rounds is $\mathsf{k}(n)$;
2. the prover messages are over the alphabet $\mathsf{a}(n)$;
3. the proof length over this alphabet is $\mathsf{l}(n)$;
4. the verifier uses $\mathsf{r}(n)$ random bits;
5. the verifier queries the prover messages in $\mathsf{q}(n)$ locations;
6. the soundness error is $\varepsilon(n)$.

Many other definitions for IPs and PCPs carry over naturally. An IOP is *public coin* if m_i is a random string and the verifier postpones any oracle queries until after receiving all the oracles from the prover (i.e., after the k-th round of interaction). An IOP is *non-adaptive* if the query locations do not depend on answers to any previous queries.

Prior work on IOPs. In prior work, [7] prove that public-coin IOPs can be compiled into non-interactive proofs in the random oracle model; their compiler is as a generalization of the Fiat–Shamir paradigm for public-coin IPs [21, 41], and of the "CS proof" constructions of Micali [37] and Valiant [49] for PCPs. Also, [6] construct 2-round IOPs (called "duplex

PCPs" there) with unconditional zero knowledge and quasilinear proof length; in comparison, short PCPs with unconditional zero knowledge are not known. Also, [43] use IOPs to obtain doubly-efficient constant-round IPs for polynomial-time bounded-space computations. In this paper, we do not study compilers for cryptographic proofs, nor zero knowledge, nor applications to interactive proofs; instead, we focus on tradeoffs of proof length versus query complexity for IOPs.

Prior work on interactive PCPs. An interactive PCP [31] is a PCP followed by a standard IP; in particular, it is an IOP where the verifier sends an empty first message and may query only the first prover message (but must read any other prover messages in full). Prior work on interactive PCPs obtains proof length that depends on the witness size rather than computation size [31, 25], as well as unconditional zero knowledge [28]. In this paper we also study proof length but our results to not seem to extend to the more restricted setting of interactive PCPs.

1.3 Proximity and robustness

To facilitate upcoming technical discussions we briefly introduce two notions that strengthen a PCP.

- *PCPs of proximity* (PCPPs) [17, 9]. On the one hand, a PCP verifier has oracle access to a candidate proof π and only decides if $\mathcal{R}|_{\mathtt{x}} \neq \emptyset$ ($\mathtt{x} \in \mathcal{L}$) or $\mathcal{R}|_{\mathtt{x}} = \emptyset$ ($\mathtt{x} \notin \mathcal{L}$). On the other hand, a PCPP verifier has oracle access to a candidate witness $\mathtt{w}$ and proof π and decides if $\mathtt{w} \in \mathcal{R}|_{\mathtt{x}}$ or $\mathtt{w}$ is far from $\mathcal{R}|_{\mathtt{x}}$ (in particular, if $\mathcal{R}|_{\mathtt{x}} = \emptyset$, then $\mathtt{w}$ is far from $\mathcal{R}|_{\mathtt{x}}$). A quantity δ known as the *proximity parameter* specifies what "far" means: if $\mathtt{w}$ is δ-far from $\mathcal{R}|_{\mathtt{x}}$ then the PCPP verifier accepts with probability at most ε, where ε is the soundness error.
- *Robust PCPs* [9]. When $\mathcal{R}|_{\mathtt{x}} = \emptyset$, the answers to the verifier's queries are, with high probability, far from any answers that make the verifier accept. A quantity ρ known as the *robustness parameter* specifies what "far" means: if $\mathcal{R}|_{\mathtt{x}} = \emptyset$ then, with probability at least $1 - \varepsilon$, the answers are ρ-far from accepting ones.

The two above notions can also be combined, yielding the definition of a *robust PCP of proximity*.

Extension to IOPs. The notions of proximity and robustness naturally extend to IOPs; see the full version for details. For example, we say that an IOP has *proximity parameter* δ if the analogous property for PCPs of proximity holds; we can then correspondingly define the complexity class $\mathbf{IOPP}[\mathsf{k}, \mathsf{a}, \mathsf{l}, \mathsf{r}, \mathsf{q}, \varepsilon, \delta]$.

2 Results

We obtain several IOP constructions with proof length and query complexity that are not known to be achievable either via PCPs or IPs alone (or even via interactive PCPs [31]). First, we show that for circuit satisfiability we can obtain IOPs with linear proof length and constant query complexity; constant round complexity and public coins suffice.

▶ **Theorem 1** (informal). *Let $\mathcal{R}$ be the relation consisting of instance-witness pairs (ϕ, w) where ϕ is a boolean circuit (of two-input NAND gates) and w is a binary input that satisfies ϕ; we use n to denote the number of gates in ϕ. There exists $a > 0$ and a public-coin IOP*

system that puts $\mathcal{R}$ in the complexity class

$$\mathbf{IOP}\left[\begin{array}{llcl} \text{rounds} & \mathsf{k}(n) & = & 3 \\ \text{answer alphabet} & \mathsf{a}(n) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(n) & = & a \cdot n \\ \text{query complexity} & \mathsf{q}(n) & = & a \\ \text{soundness error} & \varepsilon(n) & = & 1/2 \end{array}\right].$$

In particular, via [40]'s reduction from Turing machines to circuits, we deduce that

$$\mathbf{NTIME}(T) \subseteq \mathbf{IOP}\left[\begin{array}{llcl} \text{rounds} & \mathsf{k}(T) & = & 3 \\ \text{answer alphabet} & \mathsf{a}(T) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(T) & = & a \cdot T \log T \\ \text{query complexity} & \mathsf{q}(T) & = & a \\ \text{soundness error} & \varepsilon(T) & = & 1/2 \end{array}\right].$$

The main points of comparison of the above theorem with prior work are the following.

- For PCPs with constant query complexity, prior work achieved only quasilinear proof length [12, 16], with the "quasilinear" hiding several logarithmic factors. In comparison, we achieve linear proof length for circuit satisfiability, and $O(T \log T)$ proof length for nondeterministic T-time relations.
- Ben-Sasson et al. [10] show that there is $a > 0$ such that for every $\epsilon > 0$ there is a non-uniform PCP for circuit satisfiability with proof length $2^{a/\epsilon}n$ and query complexity n^{ϵ}; the non-uniformity comes from the use of algebraic-geometry (AG) codes with transitive automorphism groups, for which uniform families are not known. In comparison, we simultaneously achieve linear proof length and constant query complexity; moreover, we make a much "lighter" use of AG codes, which also allows us to avoid non-uniformity. Namely, we rely only on the multiplication properties of AG codes [14, 36], and do not rely on any code automorphisms. Looking ahead, this is because we do not route circuits on Cayley graphs induced by the automorphisms of the underlying code, unlike [10].

Second, we show that Reed–Solomon codes over binary fields (fields of characteristic 2) have 2-round IOPs of proximity with linear proof length and constant query complexity. Such codes are a key ingredient for constructing PCPs with quasilinear proof length [12]. Recall that a word $w\colon D \to \mathbb{F}$ is represented via $|w| = |D| \cdot \log|\mathbb{F}|$ bits.

▶ **Theorem 2** (informal). *Given a "fractional degree" $\varrho \in (0,1)$, define $\mathcal{R}$ to be the relation consisting of instance-witness pairs $((\mathbb{F}_{2^\lambda}, d), w)$ where $d \leq \varrho 2^\lambda$ and $w\colon \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ is the evaluation of a polynomial of degree less than d; we define the instance size to be λ, and note that w has $|w| = 2^\lambda \cdot \lambda$ bits. For every $\delta \in (0, \frac{1}{2}(1-\varrho))$ there exist $a > 0$ and a public-coin IOP of proximity (P, V) that puts $\mathcal{R}$ in the complexity class*

$$\mathbf{IOPP}\left[\begin{array}{llcl} \text{rounds} & \mathsf{k}(\lambda) & = & 2 \\ \text{answer alphabet} & \mathsf{a}(\lambda) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(\lambda) & = & a \cdot 2^\lambda \cdot \lambda \\ \text{query complexity} & \mathsf{q}(\lambda) & = & a \\ \text{soundness error} & \varepsilon(\lambda) & = & 1/2 \\ \text{proximity parameter} & \delta(\lambda) & = & \delta \end{array}\right].$$

More generally, our result concerns *additive Reed–Solomon codes*, where the domain of a codeword is a λ-dimensional affine subspace S of a potentially larger binary field $\mathbb{F}$; in such cases the above statement involves more parameters but achieves the same asymptotics. The

main point of comparison of the above theorem with prior work is [12, 16], who achieve PCPs of proximity with the same parameters but superlinear proof length: $a \cdot 2^{\lambda} \cdot \lambda \cdot \mathrm{poly}(\lambda)$.

Third, we show that tensor product codes have 1-round IOPs of proximity with sublinear proof length and constant query complexity. Given a positive integer m and linear code C with domain D and alphabet $\mathbb{F}$, the tensor product code $C^{\otimes m}$ is the linear code that comprises all functions $w\colon D^m \to \mathbb{F}$ whose restriction to any axis-parallel line is in C; the message length, block length, and distance of $C^{\otimes m}$ are each the m-th power of the corresponding parameters of C. Tensor product codes are a large family, and they include Reed–Muller codes (at least when considering the definition that bounds the variables' individual degrees, which we do, as opposed to the one that bounds their sum).

▶ **Theorem 3** (informal). *Let $m \geq 3$ and C be a linear code with domain D, alphabet $\mathbb{F}$, and relative distance τ; let $\ell := |D|$ be the block length. Define $\mathcal{R}$ to be the relation of instance-witness pairs $\big((C, m), w\big)$ such that $w \in C^{\otimes m}$; note that w has $|w| = \ell^m \cdot \log|\mathbb{F}|$ bits. For every $\delta \in (0, \frac{1}{2}\tau^m)$ there exist $a > 0$ and a public-coin IOPP system (P, V) that puts $\mathcal{R}$ in the complexity class*

$$\mathbf{IOPP}\left[\begin{array}{lccl} \text{rounds} & \mathsf{k}(\ell^m) & = & 1 \\ \text{answer alphabet} & \mathsf{a}(\ell^m) & = & \mathbb{F}_2 \\ \text{proof length} & \mathsf{l}(\ell^m) & = & o(\ell^m \cdot \log|\mathbb{F}|) \\ \text{query complexity} & \mathsf{q}(\ell^m) & = & a \\ \text{soundness error} & \varepsilon(\ell^m) & = & 1/2 \\ \text{proximity parameter} & \delta(\ell^m) & = & \delta \end{array}\right].$$

The main points of comparison of the above theorem with prior work are the following.

- Ben-Sasson and Sudan [11] and Viderman [51] give local testers for all tensor product codes with query complexity $\mathsf{q}(\ell^m) = \ell^2$; Dinur et al. [18] give local testers with $\mathsf{q}(\ell^m) = \ell$ for certain tensor product codes. In contrast, we achieve constant query complexity, with only sublinear proof length, for all tensor product codes. Moreover, given additional mild conditions, we obtain constant soundness error *even for non-constant m*.
- The work of [12, 16] implies PCPs of proximity for tensor product codes with superlinear proof length and constant query complexity. In contrast, we obtain sublinear proof length, with a single round of interaction.

Analogously to [51], we can invoke Theorem 3 on different choices of linear codes so to derive different code families that have good properties and an IOP tester (instead of a local tester as in [51]). For example, we can choose a family of linear codes with arbitrarily high rate, constant relative distance, linear-time encoding, and linear-time decoding from a constant fraction of errors [48, 29, 44]; our theorem implies a code with the same properties that *also* has a 1-round IOP of proximity with sublinear proof length and constant query complexity (cf. [51, Section 3.1]).

Similar statements hold for list-decodable codes with good parameters [27] (cf. [51, Section 3.2]); and also for locally correctable and, more generally, locally decodable codes with good parameters [52, 50, 19, 33, 32] (cf. [51, Section 3.3]). In each of these cases, the tensor product operation preserves the "key" properties of the choice of underlying code C, while endowing the resulting code with an IOP of proximity.

We obtain the above results via techniques of independent interest: we prove that, in the IOP model, there are more efficient analogues of tools that are fundamental to constructing PCPs and IPs. We now discuss these techniques.

3 Techniques

Recall that IOPs generalize both IPs, by treating the prover's messages as oracle strings, and PCPs, by allowing for multiple rounds of interaction; they also generalize interactive PCPs [31]. We prove that IOPs can express two fundamental techniques in a more efficient way than in these prior models:

(i) in *interactive proof composition*, the prover is more efficient than in PCP proof composition; and

(ii) in *sublinear sumchecks*, the verifier is more efficient than in IP sumcheck protocols.

We now discuss both of our new tools, and then how we use them.

3.1 Interactive proof composition

Proof composition [3] is used to reduce PCP query complexity, cf. [2, 30, 9]; it involves two PCPs: an outer one and an inner one. One should think of the outer proof system as having short proofs but large query complexity, while the inner proof system has long proofs but small query complexity.

The composed prover uses the outer prover to send a PCP to the composed verifier, who does not run the outer verifier but, instead, uses the inner verifier to check that the outer verifier would have accepted had it made its queries to the PCP. The composed verifier also needs an auxiliary sub-PCP for the claim that the outer verifier would have accepted; in fact, he needs one sub-PCP for each possible random string of the outer verifier. Hence, the composed prover also sends all of these sub-PCPs along with the first PCP. The benefit is that the query complexity of the composed verifier equals that of the inner verifier, which is typically verifying a much smaller statement than the outer verifier.

Beyond query complexity, most other parameters of the composed proof system are simply the sum of corresponding parameters of the outer and inner proof systems. An exception is the proof length l: it does not simply equal the sum $\mathsf{l}_{\mathsf{out}} + \mathsf{l}_{\mathsf{in}}$, but instead equals $\mathsf{l}_{\mathsf{out}} + 2^{\mathsf{r}_{\mathsf{out}}} \cdot \mathsf{l}_{\mathsf{in}}$, because the composed prover uses the inner proof system to generate a proof *for each choice of randomness of the outer proof system*. (The same is true for prover running time.)

We prove an **Interactive Proof Composition Theorem** that avoids the above limitations. The outer proof system is a robust PCP $(P_{\mathsf{out}}, V_{\mathsf{out}})$ for a relation $\mathcal{R}$, while the inner one is a k-round IOP $(P_{\mathsf{in}}, V_{\mathsf{in}})$ for V_{out}'s relation; the composed proof system is a $(\mathsf{k}+1)$-round IOP (P, V) for $\mathcal{R}$. The parameters of the composed proof system are exactly as before, except that now the new proof length is *much smaller*: $\mathsf{l}_{\mathsf{out}} + \mathsf{l}_{\mathsf{in}}$. (Ditto for the prover running time.) The crucial observation is that, after the prover sends the outer proof to the verifier, *soundness is not harmed if the verifier tells the prover his choice of outer randomness*; hence, the prover does not have to invest work for all randomness choices but can simply invest work only for the outer randomness that was chosen, which he now knows.

▶ **Theorem 4** (Interactive Proof Composition – informal). *Suppose that the relation $\mathcal{R}$ satisfies the following:*

(1) there exists a robust PCPP system $(P_{\mathsf{out}}, V_{\mathsf{out}})$ that puts $\mathcal{R}$ in the complexity class

$$\textbf{PCPP}\left[\begin{array}{ll} \text{proof length} & \mathsf{l}_{\mathsf{out}} \\ \text{randomness} & \mathsf{r}_{\mathsf{out}} \\ \text{query complexity} & \mathsf{q}_{\mathsf{out}} \\ \text{soundness error} & \varepsilon_{\mathsf{out}} \\ \text{proximity parameter} & \delta_{\mathsf{out}} \\ \text{robustness parameter} & \rho_{\mathsf{out}} \end{array}\right]$$

and

(2) for every $\mathtt{x}$ there exists an IOPP system $(P_{\mathsf{in}}, V_{\mathsf{in}})$ that puts V_{out}'s relation in the complexity class

$$\textbf{IOPP}\left[\begin{array}{ll} \text{rounds} & \mathsf{k}_{\mathsf{in}} \\ \text{proof length} & \mathsf{l}_{\mathsf{in}} \\ \text{randomness} & \mathsf{r}_{\mathsf{in}} \\ \text{query complexity} & \mathsf{q}_{\mathsf{in}} \\ \text{soundness error} & \varepsilon_{\mathsf{in}} \\ \text{proximity parameter} & \delta_{\mathsf{in}} \end{array}\right]$$

If $\delta_{\mathsf{in}} \leq \rho_{\mathsf{out}}$ then there exists an IOPP system (P, V) that puts $\mathcal{R}$ in the complexity class

$$\textbf{IOPP}\left[\begin{array}{lcl} \text{rounds} & \mathsf{k} & = 1 + \mathsf{k}_{\mathsf{in}} \\ \text{proof length} & \mathsf{l} & = \mathsf{l}_{\mathsf{out}} + \mathsf{l}_{\mathsf{in}} \\ \text{randomness} & \mathsf{r} & = \mathsf{r}_{\mathsf{out}} + \mathsf{r}_{\mathsf{in}} \\ \text{query complexity} & \mathsf{q} & = \mathsf{q}_{\mathsf{in}} \\ \text{soundness error} & \varepsilon & = \varepsilon_{\mathsf{out}} + \varepsilon_{\mathsf{in}} \\ \text{proximity parameter} & \delta & = \delta_{\mathsf{out}} \end{array}\right].$$

The above discussion and informal theorem statement omit many technical details that already arise in non-interactive proof composition (e.g., see lengthy discussions in [9, 8]), and we also need to deal with. For instance, one has to clarify the size of the sub-claim on which the the inner proof system is invoked; also, one has to carefully define the notion of a verifier to allow for the composed verifier's running time to be *smaller* than the outer verifier's query complexity. For more details, see the full version.

3.2 Sublinear sumcheck

The *sumcheck protocol* [34, 46] is an interactive proof for the claim "$\sum_{\vec{\alpha} \in H^m} w(\vec{\alpha}) = 0$", where w is the evaluation on $\mathbb{F}^m$ of an m-variate polynomial of individual degree d and H is a subset of $\mathbb{F}$. More generally, w may be a codeword in the tensor product code $C^{\otimes m}$, for a given linear code C with domain D and alphabet $\mathbb{F}$, and H is a subset of D [36]. The prover receives H and w as input, while the verifier receives H as input and w as an oracle. The protocol has m rounds and, if C has relative distance τ, the protocol has soundness error $1 - \tau^m$; also, the prover runs in time $\text{poly}(\ell^m)$, and the verifier in time $\text{poly}(\ell + m)$, where $\ell := |D|$ is C's block length.

In each round, the verifier receives a codeword w_i in C and checks that $\sum_{\alpha \in H} w_i(\alpha)$ equals a certain value γ_{i-1} determined in the previous round; in particular, the verifier reads $\Omega(\ell)$ bits. We show that the verifier complexity can be *sublinear* in ℓ, if the prover and verifier engage in an IOP instead of an IP. The intuition to "go sublinear" is simple: instead of doing these checks explicitly, the verifier uses proximity testers for doing so. Thus, in each round, the prover sends to the verifier two oracles: the codeword in w_i, and a proximity proof attesting that $w_i \in C$ and that $\sum_{\alpha \in H} w_i(\alpha) = \gamma_{i-1}$. The use of proximity proofs complicates the soundness analysis because the verifier only sees noisy codewords, but the backbone of the proof follows that of the standard sumcheck protocol. Overall, we obtain a sumcheck IOP protocol that enables a verifier to efficiently check sumchecks for codes of much larger blocklength than what he can afford in the standard sumcheck protocol.

We state our **Sublinear Sumcheck Theorem** below as a reduction: given a PCP of proximity $(P_{\text{SC}}, V_{\text{SC}})$ for subcodes of the form $C|_{H,\gamma} := \{w \in C \text{ s.t. } \sum_{\alpha \in H} w(\alpha) = \gamma\}$, we

construct an IOP of proximity (P, V) for sumchecks over H^m for $C^{\otimes m}$. The complexity of the PCPP verifier V_{SC} determines the complexity of the resulting IOPP verifier V; e.g., if the former is sublinear in C's block length ℓ, so is the latter.

▶ **Theorem 5** (Sublinear Sumcheck – informal). *Let m be a positive integer, and C a linear code with relative distance τ and block length ℓ. Suppose that there is a PCP of proximity for subcodes of the form $C|_{H,\gamma} := \{w \in C \text{ s.t. } \sum_{\alpha \in H} w(\alpha) = \gamma\}$ with proof length l_{SC}, query complexity q_{SC}, soundness error $\varepsilon_{\mathrm{SC}}$, proximity parameter δ_{SC}, prover running time $\mathsf{tp}_{\mathrm{SC}}$, and verifier running time $\mathsf{tv}_{\mathrm{SC}}$. Then there is a public-coin IOP for sumchecks over H^m for $C^{\otimes m}$ with the following parameters:*

$$\mathbf{IOP}\left[\begin{array}{llll} \text{rounds} & \mathsf{k} & = & m \\ \text{proof length} & \mathsf{l} & = & m \cdot \mathsf{l}_{\mathrm{SC}} + m \cdot \ell \\ \text{query complexity} & \mathsf{q} & = & m \cdot \mathsf{q}_{\mathrm{SC}} + m + 1 \\ \text{soundness error} & \varepsilon & = & 1 - \tau^m + \left(\varepsilon_{\mathrm{SC}} + m \cdot \delta_{\mathrm{SC}}\right) \\ \text{prover time} & \mathsf{tp} & = & m \cdot \mathsf{tp}_{\mathrm{SC}} + m \cdot \ell^m \\ \text{verifier time} & \mathsf{tv} & = & m \cdot \mathsf{tv}_{\mathrm{SC}} + O(m) \end{array}\right].$$

In later sections, it is more natural to state the theorem without assuming that w is a codeword in $C^{\otimes m}$, so the reduction also takes as input a PCP of proximity $(P_{\otimes}, V_{\otimes})$ for $C^{\otimes m}$ that is invoked on w; this introduces additional terms in the parameters. More generally, both of the PCPs of proximity $(P_{\mathrm{SC}}, V_{\mathrm{SC}})$ and $(P_{\otimes}, V_{\otimes})$ can in fact be IOPs of proximity, and we state our theorem for this more general case, which we need. For more details, see the full version.

3.3 Applying the new tools

We now sketch how we use the above new tools to derive the results of Section 2. We begin by discussing our results on proximity testing to codes (stated later); we then turn to circuit satisfiability (stated earlier) because its proof requires one of these results on proximity testing.

Intuition behind Theorem 2. The construction of linear-size IOPs of proximity for Reed–Solomon codes over binary fields follows from one invocation of our Interactive Proof Composition Theorem with [12]'s robust PCPs of proximity for Reed–Solomon codes as the outer proof system, and [38]'s PCPs of proximity for nondeterministic languages as the inner proof system. Informally, in the first round, the prover sends to the verifier a [12] PCP of proximity, which reduces proximity testing of codewords over $\mathbb{F}_{2^\lambda}$ to proximity testing of sub-codewords over $\mathbb{F}_{2^{\lambda/2}+O(1)}$ with only constant overheads; in the second round, the verifier sends his choice of outer randomness, and the prover replies with a [38] PCP of proximity for the sub-codeword. The proof length of this latter component is quasilinear, but is applied to a claim of "square-root size" only, so we obtain linear proof length.

Intuition behind Theorem 3. The construction of sublinear-size IOPs of proximity for tensor product codes follows from one invocation of our Interactive Proof Composition Theorem with [11, 51]'s robust local tester for tensor product codes as the outer proof system, and [38]'s PCPs of proximity for nondeterministic languages as the inner proof system. Unlike before, we now use one round, because the outer proof system only relies on a local tester rather than a PCP of proximity. The verifier thus simply sends his choice of outer randomness, and the prover replies with a [38] PCP of proximity for a suitable sublinear-size

sub-codeword. Since the proof length of this latter component is quasilinear but is applied to a sublinear-size claim, we obtain sublinear proof length.

A summary: overall, we can summarize the above sketches via the following diagram of implications.

Theorem 2 linear-size IOPP for Reed–Solomon codes	⟵	Theorem 4 interactive proof composition	+	[12] robust PCPs of proximity for Reed–Solomon codes	+	[38] PCP of proximity for **NTIME**
Theorem 3 sublinear-size IOPP for tensor product codes	⟵	Theorem 4 interactive proof composition	+	[11, 51] robust local testing for tensor product codes	+	[38] PCP of proximity for **NTIME**

Intuition behind Theorem 1. We now turn to how to construct 3-round IOPs for circuit satisfiability with linear proof length and constant query complexity.

The first step of many PCP constructions is to arithmetize the NP statement at hand (in our case, the satisfiability of a boolean circuit) by reducing it to a "PCP-friendly" problem that looks like a constraint satisfaction problem over a well-chosen graph and whose assignments involve codewords in a well-chosen linear code C. Meir observes in [35, 36] that key features of C are good relative distance and, moreover, a *multiplication property*: coordinate-wise multiplication of codewords yields codewords in a code whose relative distance is still good [14, 36]. Moreover, to obtain short PCPs, the aforementioned graph is typically chosen so to behave like a routing network [42]; for example, [12] use De Bruijn graphs, while [10] use hypercubes. To support such graphs, the automorphism group of C has to be rich enough. This typically holds for Reed–Solomon codes [12] which have a doubly-transitive automorphism group, but is a significantly harder condition to fulfill for AG codes [10], for which obtaining a transitive automorphism group is quite involved and, currently, can only be achieved non-uniformly.

The aforementioned first step would be problematic in our setting, because known routing techniques introduce either logarithmic overheads (as in [12]) or large query complexity (as in [10]), so it is not clear how we could use them. Departing from these prior works, we do not rely on any routing, and instead immediately leverage one round of interaction to directly reduce circuit satisfiability to a sumcheck instance over a given linear code C. Also, we only assume that C has good relative distance and a multiplication property [14], *but we do not rely on any automorphisms.*

Informally, the prover first sends three codewords w_1, w_2, w_3 over a field $\mathbb{F}$; the first codeword encodes values of the left wires of all gates, the second encodes values for the right wires of all gates, and the third encodes values for the output wires of all gates. (When a gate has fan-out greater than 1 we still consider 1 output wire.) The verifier now must check several things. First, that wire values are boolean and the output gate wire equals 0. Second, that the wire values are "locally consistent" with each gate: for every $i \in [n]$, $w_3(i)$ is the NAND of $w_1(i)$ and $w_2(i)$. Third, that the three encodings of wire values are consistent with the circuit topology: namely, if $\ell(i)$ represents the left wire used to compute i, and $r(i)$ represents the right wire used to compute i, the topology requires that $w_3(\ell(i)) = w_1(i)$ and $w_3(r(i)) = w_2(i)$ for every i. The verifier cannot directly conduct these checks (as doing so would incur linear query complexity); instead, the verifier sends some randomness to the prover so to "bundle" the checks into one sumcheck.

But how should the verifier sample randomness to achieve this bundling? One option is to sample a random element in $\mathbb{F}$ per check so to construct a random subset sum, which can be viewed as an n-variate polynomial of total degree 1, whose coefficients are the checks,

evaluated at a random point. If not all checks are satisfied, the polynomial is non-zero, and its random evaluation cannot attain any value with too large probability. However, constructing a random subset sum is inefficient because the verifier samples and sends to the prover $\Omega(n)$ random bits, in order to describe the random point. Nevertheless, the verifier may hope to do better by using a *different* low-degree polynomial for the bundling. In general, if the polynomial has m variables each of degree at most d, the verifier must sample and send m field elements; this preserves soundness provided that $|\mathbb{F}| = \Omega(md)$ (for a constant probability of avoiding any particular output value by the Schwartz–Zippel Lemma [45, 53, 15]) and $d^m = \Omega(n)$ (to bundle all checks). For example, the univariate case of $m = 1$ was considered in [5] when reducing to a sumcheck problem; the multivariate case of $m = \log n$ or $m = \frac{\log n}{\log \log n}$ was considered in later works. Unfortunately, either setting does not work for *constant-size* fields, which we ultimately use to obtain linear proof length.

Taking a step back from polynomials, we see that all we need is an *evading set* S for $\mathbb{F}^n$, which is a small set such that for any non-zero $v \in \mathbb{F}^n$ the inner product $\langle r, v \rangle$, for random $r \in S$, does not attain any particular value $a \in \mathbb{F}$ with too high probability. Good constructions of evading sets are known: they relax a well-studied notion called ϵ-biased sets [39]. In particular, results of [1] imply that, for any ϵ, $\mathbb{F}^n$ has an evading set S of size $\text{poly}(\frac{n}{\epsilon})$ and the aforementioned probability is $\gamma := \epsilon + \frac{1}{|\mathbb{F}|}$; in particular, such a construction is suitable for constant-size fields.

Below we informally state the reduction (see the full version for details), using the following notion: we say that a linear code C' is a *degree d-closure* of C if, for every $w_1, \ldots, w_m \in C$ and m-variate polynomial P of total degree at most d, it holds that $w' \in C'$ where the i-th entry of w' is the evaluation of P on the i-th coordinates of $w_1, \ldots, w_m$.

▶ **Lemma 6** (Circuit SAT to Sumcheck – informal). *Let n be a positive integer, $C \subseteq \mathbb{F}^D$ an n-systematic linear code, ϕ an n-gate boolean circuit (of two-input NAND gates), and S an evading set for $\mathbb{F}^n$. There is a 1-round IOP that reduces satisfiability of ϕ to proximity testing to C and a sumcheck over any degree-3 closure of C. Moreover, the IOP introduces only constant overheads in all relevant parameters, including proof length and query complexity.*

After reducing circuit satisfiability to sumcheck over the given code C, we are left to choose C so to ensure that the sumcheck can be carried out with 2 additional rounds, linear proof length, and constant query complexity.

For this, our starting point is [23, 47]'s efficient construction of a code family with constant rate, relative distance, and alphabet size. Note that since these codes are AG codes, they have a naturally-defined degree-3 closure. Also, their construction is uniform, and thus represents a much "lighter" use of AG codes as compared to in [10].

If we simply choose C to be a code from this AG code family, then it is not clear how to efficiently conduct the sumcheck. However, what does work is to take C to be the tensor product of $O(1)$ copies of this AG code. Informally, in this way, we can invoke our Sublinear Sumcheck Theorem (Theorem 5) on the tensor product code C and we can test proximity to it by Theorem 3. See the full version for details.

Overall, we can summarize the above sketch via the following diagram of implications.

$$\underset{\substack{\text{linear-size IOP}\\ \text{for circuit SAT}}}{\textbf{Theorem 1}} \longleftarrow \underset{\substack{\text{from circuit SAT}\\ \text{to sumcheck}}}{\text{Lemma 6}} + \underset{\substack{\text{sublinear}\\ \text{sumcheck}}}{\text{Theorem 5}} + \underset{\substack{\text{sublinear-size IOP}\\ \text{for tensor product codes}}}{\text{Theorem 3}} + \underset{\substack{\text{efficient construction}\\ \text{of AG codes}}}{[23, 47]}$$

4 Open questions

The question of whether there exist PCPs with linear proof length and constant query complexity remains open. Nevertheless, our work suggests additional questions that may be stepping stones in this and other intriguing directions:

1. Is there a *one*-round IOP for circuit satisfiability with linear proof length and query complexity? (Our IOP for circuit satisfiability requires 3 rounds.)
2. Is there an IOP for $\mathbf{NTIME}(T)$ with linear proof length and query complexity, for *some* number of rounds? (Our results, like [10], only imply proof length $O(T \log T)$.)
3. Is there an IOP for *succinct* circuit satisfiability with linear proof length and query complexity? (Our results, like [10], "stop" at **NP** but do not extend to **NEXP**.)

Finally, while "positive" applications of IOPs are known (e.g., non-interactive proofs in the random oracle model [7]), "negative" ones are not: do IOP constructions with good parameters imply inapproximability results that are not known to be implied by known PCP constructions?

References

1 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS'92.

3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in FOCS'92.

4 László Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC'85, pages 421–429, 1985.

5 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC'91, pages 21–32, 1991.

6 Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasilinear-size zero knowledge from linear-algebraic PCPs. In *Proceedings of the 13th Theory of Cryptography Conference*, TCC'16-A, pages 33–64, 2016.

7 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Proceedings of the 14th Theory of Cryptography Conference*, TCC'16-B, pages 31–60, 2016.

8 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, CCC'05, pages 120–134, 2005.

9 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

10 Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant rate PCPs for Circuit-SAT with sublinear query complexity. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'13, pages 320–329, 2013.

11 Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Structures and Algorithms*, 28(4):387–402, 2006.

12 Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. Preliminary version appeared in STOC'05.

13 Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, STOC'03, pages 612–621, 2003.

14 D. V. Chudnovsky and G. V. Chudnovsky. Algebraic complexities and algebraic curves over finite fields. *Journal of Complexity*, 4(4):285–316, 1988.

15 Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

16 Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

17 Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'04, pages 155–164, 2004.

18 Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of LDPC codes. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, and of the 10th International Workshop on Randomization and Computation*, APPROX-RANDOM'06, pages 304–315, 2006.

19 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012. Preliminary version appeared in STOC'09.

20 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS'91, pages 2–12, 1991.

21 Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO'86, pages 186–194, 1986.

22 Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. In *Theoretical Computer Science*, pages 156–161, 1988.

23 Arnaldo Garcia and Henning Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248–273, 1996.

24 Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, 53:558–655, July 2006. Preliminary version in STOC'02.

25 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for Muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC'08, pages 113–122, 2008.

26 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version appeared in STOC'85.

27 Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. List decoding tensor products and interleaved codes. *SIAM Journal on Computing*, 40(5):1432–1462, 2011. Preliminary version appeared in STOC'09.

28 Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, pages 173–190, 2010.

29 Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005. Preliminary version appeared in STOC'03.

30 Prahladh Harsha and Madhu Sudan. Small PCPs with low query complexity. *Computational Complexity*, 9(3–4):157–201, Dec 2000. Preliminary version in STACS'01.

31 Yael Kalai and Ran Raz. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP'08, pages 536–547, 2008.

32 Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally-correctable and locally-testable codes with sub-polynomial query complexity. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC'16, pages 202–215, 2016.

33 Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of the ACM*, 61(5):28:1–28:20, 2014. Preliminary version appeared in STOC'11.

34 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

35 Or Meir. Combinatorial PCPs with short proofs. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity*, CCC'12, 2012.

36 Or Meir. IP = PSPACE using error-correcting codes. *SIAM Journal on Computing*, 42(1):380–403, 2013.

37 Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Preliminary version appeared in FOCS'94.

38 Thilo Mie. Short PCPPs verifiable in polylogarithmic time with o(1) queries. *Annals of Mathematics and Artificial Intelligence*, 56:313–338, 2009.

39 Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC'90, pages 213–223, 1990.

40 Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.

41 David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Proceedings of the 14th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'96, pages 387–398, 1996.

42 Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, STOC'94, pages 194–203, 1994.

43 Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC'16, pages 49–62, 2016.

44 Ron M. Roth and Vitaly Skachek. Improved nearly-MDS expander codes. *IEEE Transactions on Information Theory*, 52(8):3650–3661, 2006.

45 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

46 Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

47 Kenneth W. Shum, Ilia Aleshnikov, P. Vijay Kumar, Henning Stichtenoth, and Vinay Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert–Varshamov bound. *IEEE Transactions on Information Theory*, 47(6):2225–2241, 2001.

48 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996. Preliminary version appeared in STOC'95.

49 Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, TCC'08, pages 1–18, 2008.

50 Michael Viderman. A note on high-rate locally testable codes with sublinear query complexity, 2010. ECCC TR10-171.

51 Michael Viderman. A combination of testability and decodability by tensor products. *Random Structures and Algorithms*, 46(3):572–598, 2015. Preliminary version appeared in APPROX-RANDOM'12.

52 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 55(1), 2008. Preliminary version appeared in STOC'07.

53 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the 1979 International Symposium on Symbolic and Algebraic Computation*, EUROSAM'79, pages 216–226, 1979.

Dynamic Parameterized Problems and Algorithms*†

Josh Alman[1], Matthias Mnich[2], and Virginia Vassilevska Williams[3]

1 **MIT CSAIL, Cambridge, MA, USA**
jalman@mit.edu
2 **Universität Bonn, Institut für Informatik, Bonn, Germany; and Maastricht University, Department of Quantitative Economics, Maastricht, The Netherlands**
mmnich@uni-bonn.de
m.mnich@maastrichtuniversity.nl
3 **MIT CSAIL, Cambridge, MA, USA**
virgi@mit.edu

Abstract

Fixed-parameter algorithms and kernelization are two powerful methods to solve NP-hard problems. Yet, so far those algorithms have been largely restricted to *static* inputs.

In this paper we provide fixed-parameter algorithms and kernelizations for fundamental NP-hard problems with *dynamic* inputs. We consider a variety of parameterized graph and hitting set problems which are known to have $f(k)n^{1+o(1)}$ time algorithms on inputs of size n, and we consider the question of whether there is a data structure that supports small updates (such as edge/vertex/set/element insertions and deletions) with an update time of $g(k)n^{o(1)}$; such an update time would be essentially optimal. Update and query times independent of n are particularly desirable. Among many other results, we show that Feedback Vertex Set and k-Path admit dynamic algorithms with $f(k)\log^{O(1)} n$ update and query times for some function f depending on the solution size k only.

We complement our positive results by several conditional and unconditional lower bounds. For example, we show that unlike their undirected counterparts, Directed Feedback Vertex Set and Directed k-Path do not admit dynamic algorithms with $n^{o(1)}$ update and query times even for constant solution sizes $k \leq 3$, assuming popular hardness hypotheses. We also show that unconditionally, in the cell probe model, Directed Feedback Vertex Set cannot be solved with update time that is purely a function of k.

1998 ACM Subject Classification F2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Dynamic algorithms, fixed-parameter algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.41

1 Introduction

The area of dynamic algorithms studies data structures that store a dynamically changing instance of a problem, can answer queries about the current instance and can perform small changes on it. The major question in this area is, how fast can updates and queries be?

* A full version of the paper is available at https://arxiv.org/abs/1707.00362.

† J.A. is supported by NSF Grant DGE-114747. M.M. is supported by ERC Starting Grant 306465 (BeyondWorstCase). V.V.W. is supported by NSF Grants CCF-141-7238, CCF-1528078 and CCF-1514339, and BSF Grant BSF:2012338. This work was initiated while J.A. and V.V.W. were at Stanford University.

The most studied dynamic problems are dynamic graph problems such as connectivity (e.g., [45, 47, 48, 67]), reachability [41], shortest paths (e.g., [7, 26, 42]), and maximum matching [8, 39, 66]. For a dynamic graph algorithm, the updates are usually edge or vertex insertions and deletions. Any dynamic graph algorithm that can perform edge insertions can be used for a static algorithm by starting with an empty graph, and using m insertions to insert the m-edge input graph. That is, if the update time of the dynamic algorithm is $u(m)$ then the static problem can be solved in $O(m \cdot u(m))$ time, plus the time to query for the output. Hence, if a problem requires $\Omega(f(m))$ time to be solved statically, then any dynamic algorithm that can insert edges, and can be queried for the problem solution in $o(f(m))$ time, must need $\Omega(f(m)/m)$ (amortized) time to perform updates. This is not limited to edge updates; similar statements are true for vertex insertions and other update types. A fundamental question is which problems can be fully dynamized, i.e., have dynamic algorithms supporting updates in $O(f(m)/m)$ time where $f(m)$ is the static runtime?

This question is particularly interesting for static problems that can be solved in near-linear time. For them, we are interested in near-constant time updates—the holy grail of dynamic algorithms. The field of dynamic algorithms has achieved such full dynamization for many problems. A prime example of the successes of this vibrant research area is the *dynamic connectivity* problem: maintaining the connected components of a graph under edge updates, to answer queries about whether a pair of vertices is connected. This problem can be solved with amortized expected update time $O(\log n \log\log^2 n)$ [48, 67] and query time $O(\log n/\log\log\log n)$; polylogarithmic deterministic amortized bounds are also known, the current best by Wulff-Nielsen [71]. After much intense research on the topic [44, 46, 47], the first polylogarithmic *worst case expected* update times were obtained by Kapron et al. [53], who were the first to break through what seemed like an $\Omega(\sqrt{n})$ barrier; the bounds of Kapron et al. [53] were recently improved by Gibb et al. [37]. Similar $\tilde{O}(1)$ update and query time bounds[1] are known for many problems solvable in linear time such as dynamic minimum spanning tree, biconnectivity and 2-edge connectivity [45, 47], and maximal matching [5, 66].

Barriers for dynamization have also been studied extensively. Many unconditional, cell probe lower bounds are known. For instance, for connectivity and related problems it is known [64, 65] that either the query time or the update time needs to be $\Omega(\log n)$. However, current cell probe lower bound techniques seem to be limited to proving polylogarithmic lower bounds. In contrast, conditional lower bounds based on popular hardness hypotheses have been successful at giving tight bounds for problems such as dynamic reachability, dynamic strongly connected components and many more [1, 43, 54, 62].

While the field of dynamic algorithms is very developed, practically all the problems which have been studied are polynomial-time solvable problems. What about NP-hard problems? Do they have fast dynamic algorithms? By the discussion above, it seems clear that (unless $\mathsf{P} = \mathsf{NP}$), superpolynomial query/update times are necessary, and surely this is not as interesting as achieving near-constant time updates. If the problem is relaxed, and instead of exact solutions, approximation algorithms are sufficient, then efficient dynamic algorithms have been obtained for some polynomial time approximable problems such as dynamic approximate vertex cover [5, 8, 61]. What if we insist on exact solutions?

The efficient dynamization question does make sense for *parameterized* NP-hard problems. For such problems, each instance is measured by its size n as well as a *parameter* k that measures the optimal solution size, the treewidth or genus of the input graph, or any similar structural property. If $\mathsf{P} \neq \mathsf{NP}$, then the runtime of any algorithm for such a problem needs

[1] Throughout this paper, we write $\tilde{O}(f(n,k))$ to hide polylog(n) factors.

to be superpolynomial, but it is desirable that the superpolynomiality is only in terms of k. That is, one searches for so-called *fixed-parameter algorithms* with runtime $f(k) \cdot n^c$ for some computable function f and some fixed constant c independent of k and n. The *holy grail* here is an algorithm with runtime $f(k) \cdot n$ where f is a modestly growing function. Such linear-time fixed-parameter algorithms can be very practical for small k. The very active area of fixed-parameter algorithms has produced a plethora of such algorithms for many different parameterized problems. Some examples include (1) many branching tree algorithms such as those for Vertex Cover and d-Hitting Set, (2) many algorithms based on color-coding [4] such as for k-Path, (3) all algorithms that follow from Courcelle's theorem[2] [18], and (4) many more [11, 28, 32, 52, 58, 68, 70].

We study whether NP-hard problems with (near-)linear time fixed-parameter algorithms can be made efficiently dynamic. The main questions we address are:

- Which problems solvable in $f(k) \cdot n^{1+o(1)}$ time have dynamic algorithms with update and query times at most $f(k) \cdot n^{o(1)}$?
- Which problems solvable in $f(k) \cdot n$ time have dynamic algorithms with update and query times that depend solely on k and not on n?
- Can one show that (under plausible conjectures) a problem requires $\Omega(f(k) \cdot n^{\delta})$ (for constant $\delta > 0$) update time to maintain dynamically even though statically it can be solved in $f(k) \cdot n^{1+o(1)}$ time?

1.1 Prior Work

We are aware of only a handful papers related to the question that we study. Bodlaender [9] showed how to maintain a tree decomposition of constant treewidth under edge and vertex insertions and deletions with $O(\log n)$ update time, as long as the underlying graph always has treewidth at most 2. Dvořák et al. [29] obtained a dynamic algorithm maintaining a tree-depth decomposition of a graph under the promise that the tree-depth never exceeds D; edge and vertex insertions and deletions are supported in $f(D)$ time for some function f. Dvořák and Tůma [30] obtained a dynamic data structure that can count the number of induced copies of a given h-vertex graph, under edge insertions and deletions, and if the maintained graph has bounded expansion, the update time is bounded by $O(\log^{h^2} n)$.

A more recent paper by Iwata and Oka [51] gives several dynamic algorithms for the following problems, under the promise that the solution size never grows above k: (1) an algorithm that maintains a Vertex Cover in a graph under $O(k^2)$ time edge insertions and deletions and $f(k)$ time queries[3], (2) an algorithm for Cluster Vertex Deletion under $O(k^8 + k^4 \log n)$ time edge updates and $f(k)$ time queries[4], and (3) an algorithm for Feedback Vertex Set in graphs with maximum degree Δ where edge insertions and deletions are supported in amortized time $2^{O(k)}\Delta^3 \log n$. Notably, when discussing Feedback Vertex Set, the paper concludes: "It seems an interesting open question whether it is possible to construct an efficient dynamic graph without the degree restriction."

The final related papers are by Abu-Khzam et al. [2, 3]. Although these papers talk about parameterized problems and dynamic problems, the setting is very different. Their

[2] Courcelle's theorem states that every problem definable in monadic second-order logic of graphs can be decided in linear time on graphs of bounded treewidth.

[3] Here $f(k)$ denotes the runtime of the fastest fixed-parameter algorithm for Vertex Cover when run on k-vertex graphs.

[4] Here $f(k)$ denotes the runtime of the fastest fixed-parameter algorithm for Cluster Vertex Deletion when run on k^5-vertex graphs.

problem is, given two instances I_1 and I_2 of a problem that only differ in k "edits", and a solution S_1 of I_1, to find a feasible solution S_2 of I_2 that is at Hamming distance at most d from S_1. The question of study is whether such problems admit fixed-parameter algorithms for parameters k and ℓ. That question though is not about data structures but about a single update. Moreover, their algorithm is given S_1 as input, which—unlike a dynamic data structure—cannot force the initial solution to have any useful properties. Thus, the hardness results in their setting do not translate to our data structure setting. Furthermore, the runtimes in their setting, unlike ours, must have at least a linear dependence on the size of the input, as one has to at least read the entire input.

Besides the work on parameterized dynamic algorithms, there has been some work on parameterized streaming algorithms by Chitnis et al. [17]. This work focused on Maximal Matching and Vertex Cover. The difference between streaming and dynamic algorithms is that (a) the space usage of the algorithm is the most important aspect for streaming, and (b) in streaming, a solution is only required at the end of the stream, whereas a dynamic algorithm can be queried at any point and needs to be efficient throughout, but can use a lot of space. For Vertex Cover instances whose solution size never exceeds k, Chitnis et al. [17] give a one-pass randomized streaming algorithm that uses $O(k^2)$ space and answers the final query in $2^{O(k)}$ time; when the vertex cover size can exceed k at any point, there is a one-pass randomized streaming algorithm using $O(\min\{m, nk\})$ space and answering the query in $O(\min\{m, nk\}) + 2^{O(k)}$ time.

The relevant prior work on (static) fixed-parameter algorithms [4, 6, 10, 12, 13, 14, 15, 16, 19, 20, 22, 23, 24, 27, 31, 34, 38, 40, 49, 50, 56, 57, 59, 68] is described in the appendix.

1.2 Our Contributions

Algorithmic Results. We first define the notion of a fixed parameter dynamic problem as a parameterized problem with parameter k that has a data structure supporting updates and queries to an instance of size n in time $f(k)n^{o(1)}$. The class FPD contains all such parameterized problems. By a formalization of our earlier discussion, FPD is contained in the class of parameterized problems admitting algorithms running in time $f(k)n^{1+o(1)}$. After this, we introduce two techniques for making fixed-parameter algorithms dynamic, and then use them to develop dynamic fixed-parameter algorithms for a multitude of fundamental optimization problems. Our algorithmic contributions are stated in Theorem 1 below. In the runtimes, $DC(n)$ refers to the time per update to a dynamic connectivity data structure on n vertices, which from prior work can be:

- *expected amortized* update time $O(\log n(\log\log n)^2)$, or
- *expected worst case* time $O(\log^4 n)$, or
- *deterministic amortized* time $O(\log^2 n/\log\log n)$.

Which of these bounds we pick determines the type of guarantees (expected vs. deterministic, worst case vs. amortized) that the algorithm gives.

▶ **Theorem 1.** *The following problems admit dynamic fixed-parameter algorithms:*

- Vertex Cover *parameterized by solution size under edge insertions and deletions, with* $O(1)$ *amortized or* $O(k)$ *worst case update time and* $O(1.2738^k)$ *query time,*
- Connected Vertex Cover *parameterized by solution size under edge insertions and deletions, with* $O(k2^k)$ *update time and* $O(4^k)$ *query time,*
- d-Hitting Set *for all values of* d *parameterized by solution size under set insertions and deletions, either with* $O(kd^k)$ expected *update time and* $O(k)$ *query time, or with*

$O(f(k,d))$ *(worst-case, deterministic) update time and* $O(d^k d!(k+1)^d)$ *query time, for some function* f *loosely bounded by* $(d!)^d k^{O(d^2)}$.

- Edge Dominating Set *parameterized by solution size under edge insertions and deletions, with* $O(1)$ *update time and* $O(2.2351^k)$ *query time,*
- Feedback Vertex Set *parameterized by solution size under edge insertions and deletions, with* $2^{O(k \log k)} \log^{O(1)} n$ *amortized update time and* $O(k)$ *query time,*
- Max Leaf Spanning Tree *parameterized by solution size under edge insertions and deletions, with* $O(3.72^k + k^5 \log n + DC(n))$ *amortized update time, and maintains the current max leaf spanning tree explicitly in memory,*
- Dense Subgraph in Graphs with Degree Bounded by Δ *parameterized by the number of vertices in the subgraph under edge insertions and deletions, with* $2^{O(k\Delta)} \cdot DC(n)$ *update time and* $2^{O(k\Delta)} \log n$ *query time.*
- Undirected k-Path *parameterized by the number of vertices on the path, with* $k!2^{O(k)} \cdot DC(n)$ *update time and* $k!2^{O(k)} \log n$ *query time.*
- Edge Clique Cover *parameterized by the number of cliques and under the promise that the solution never grows bigger than* $g(k)$*, with* $O(4^{g(k)})$ *update time and* $2^{2^{O(k)}} + O(2^{4g(k)})$ *query time.*
- Point Line Cover *and* Line Point Cover *parameterized by the size of the solution and under point and line insertions and deletions, respectively, with* $O(g(k)^3)$ *update time and* $O(g(k)^{2g(k)+2})$ *query time, under the promise that the solution never grows to more than* $g(k)$.

Discussion of the Algorithmic Results. Our dynamic algorithm for Vertex Cover and that of Iwata and Oka [51] both have query time $O(1.2738^k)$, by using the best known fixed parameter algorithm for Vertex Cover on the maintained kernel. However, our algorithm improves upon theirs in two ways. First, our update time is amortized *constant* or $O(k)$ worst case, whereas the Iwata-Oka algorithm has update time $O(k^2)$. Second, their update time bound of $O(k^2)$ only holds if the vertex cover is guaranteed to never grow larger than k throughout the sequence of updates. Namely, their update time depends on the size of their maintained kernel, which may become unbounded in terms of k. Our algorithm does not need any such promise—it will always have fast (amortized $O(1)$ or $O(k)$ worst case) update time and return a vertex cover of size k if it exists, or determine that one does not. This is a much stronger guarantee.

Our dynamic algorithm and Chitnis et al.'s streaming algorithm for Vertex Cover are both based on Buss' kernel, but our algorithm is markedly different from theirs. In particular, we actually work with a modified kernel that allows us to achieve constant amortized update time. Because our algorithm is completely deterministic, it necessarily needs $\Omega(m)$ space, and our algorithm does indeed take linear space.

We give two algorithms for d-Hitting Set. The first is based on a randomized branching tree method, while the second is deterministic and maintains a small kernel for the problem. For every constant d, any d-Hitting Set instance on m sets and n elements has a kernel constructible in time $O(dn+2^d m)$ that has $O(d^{d+1}d!(k+1)^d)$ sets, due to van Bevern [68], and a kernel constructible in time $O(m)$ that has $O((k+1)^d)$ sets, due to Fafianie and Kratsch [33]. Unfortunately, it seems difficult to efficiently dynamize these kernel constructions. Because of this, we present a *novel kernel* for the problem. Our kernel can be constructed in $O(dn+3^d m)$ time and has size $O((d-1)!(k+1)^d)$. It also has nice properties that make it possible to maintain it dynamically with update time that is a function of only k and d. In fact, for any fixed d, the update time is polynomial in k.

Our algorithm for Feedback Vertex Set is a nice combination of kernelization and a branching tree. Aside from our dynamic kernel for d-Hitting Set, this is probably the most involved of our algorithms. Iwata and Oka [51] had also presented a dynamic fixed-parameter algorithm for Feedback Vertex Set. However, their update time depends linearly on the maximum degree of the graph, and is hence efficient only for bounded degree graphs. Their paper asks whether one can remove this costly dependence on the degree. Our algorithm answers their question in the affirmative—it has fast updates regardless of the graph density.

All of our algorithms, except for the last two in the theorem, meet their update and query time guarantees regardless of whether the currently stored instance has a solution of size k or not. The two exceptions, Edge Clique Cover and Point Line Cover, only work under the promise that the solution never grows bigger than a function of k. In this sense they are similar to most of the algorithms from prior work [29, 51]. There does seem to be an inherent difficulty to removing the promise requirement, however. In fact, in the parameterized complexity literature, these two problems are also exceptional, in the sense that their fastest fixed parameter algorithms run by computing a kernel and then running a brute force algorithm on it [23, 55], rather than anything more clever.

Hardness Results. In addition to the above algorithms, we also prove conditional lower bounds for several parameterized problems, showing that they are likely not in FPD. To our knowledge, ours are the first lower bounds for any dynamic parameterized problems.

The hardness hypothesis we assume concerns Reachability Oracles (ROs) for DAGs: an RO is a data structure that stores a directed acyclic graph and for any queried pair of vertices s, t, can efficiently answer whether s can reach t. (An RO does not perform updates.) Our main hypothesis is as follows:

▶ **Hypothesis 2** (RO Hypothesis). *On a word-RAM with $O(\log m)$ bit words, any Reachability Oracle for directed acyclic graphs on m edges must either use $m^{1+\varepsilon}$ preprocessing time for some $\varepsilon > 0$, or must use $\Omega(m^{\delta})$ time to answer reachability queries for some constant $\delta > 0$.*

The only known ROs either work by computing the transitive closure of the DAG during preprocessing, thus spending $\Theta(\min\{mn, n^{\omega}\})$ time (where n is the number of vertices and $2 \leq \omega < 2.373$ [36, 69]), or by running a BFS/DFS procedure after each query, thus spending $O(m)$ time. Both of these runtimes are much larger than our assumed hardness; hence the RO Hypothesis is very conservative.

We also use a slightly weaker version of the RO Hypothesis, asserting that its statement holds true even restricted to DAGs that consist of ℓ layers of vertices (for some fixed constant ℓ), so that the edges go only between adjacent layers in a fixed direction, from layer i to layer $i + 1$. While this new *LRO Hypothesis* is certainly weaker, we show that it is implied by either of two popular hardness hypotheses: the 3SUM Conjecture and the Triangle Conjecture. The former asserts that when given n integers within $\{-n^c, \ldots, n^c\}$ for some constant c, deciding whether three of them sum to 0 requires $n^{2-o(1)}$ time on a word-RAM with $O(\log n)$ bit words. The latter asserts that detecting a triangle in an m-edge graph requires $\Omega(m^{1+\varepsilon})$ time for some $\varepsilon > 0$. These two conjectures have been used for many conditional lower bounds [1, 35, 54].

Pătraşcu studied the RO Hypothesis, and while he was not able to prove it, the following strong cell probe lower bound follows from his work [63]: there are directed acyclic graphs on m edges for which any RO that uses $m^{1+o(1)}$ preprocessing time (and hence space) in the word-RAM with $O(\log n)$ bit words, must have $\omega(1)$ query time. Using this statement, unconditional, albeit weaker lower bounds can be proven as well. This is what we prove:

▶ **Theorem 3.** *Fix the word-RAM model of computation with w-bit words for $w = O(\log n)$ for inputs of size n. Assuming the LRO Hypothesis, there is some $\delta > 0$ for which the following dynamic parameterized graph problems on m-edge graphs require either $\Omega(m^{1+\delta})$ preprocessing or $\Omega(m^{\delta})$ update or query time:*

- Directed k-Path *under edge insertions and deletions,*
- Steiner Tree *under terminal activation and deactivation, and*
- Vertex Cover Above LP *under edge insertions and deletions.*

Under the RO Hypothesis (and hence also under the LRO Hypothesis), there is a $\delta > 0$ so that Directed Feedback Vertex Set *under edge insertions and deletions requires $\Omega(m^{\delta})$ update time or query time.*

Unconditionally, there is no computable function f for which a dynamic data structure for Directed Feedback Vertex Set *performs updates and answers queries in $O(f(k))$ time.*

Our lower bounds show that, although k-Path and Feedback Vertex Set have fixed parameter dynamic algorithms for undirected graphs, they probably do not for directed graphs. Interestingly, the fixed-parameter algorithms for k-Path in the static setting work similarly on both undirected and directed graphs, so there only seems to be a gap in the dynamic setting.

All problems for which we prove lower bounds have $f(k)n^{1+o(1)}$ time static algorithms, except for Vertex Cover above LP. However, it seems that the reason why the current algorithms are slower is largely due to the fact that near-linear time algorithms for maximum matching are not known. Recent impressive progress on the matching problem [60] gives hope that an $f(k)n^{1+o(1)}$ time algorithm for Vertex Cover above LP might be possible.

A common feature of most of the problems above is that they are either not known to have a polynomial kernel (like Directed Feedback Vertex Set), or do not have one unless $\mathsf{NP} \subseteq \mathsf{coNP}/poly$ (like k-Path [10] and Steiner Tree parameterized by the number of terminal pairs [27]). One might therefore conjecture that problems which cannot be made fixed parameter dynamic do not have polynomial kernels, or vice versa. Tempting as it is, this intuition turns out to be false. Vertex Cover Above LP does not have a dynamic fixed-parameter algorithm, yet it is known to admit a polynomial kernel [56]. On the other hand, the k-Path problem on undirected graphs also does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP}/poly$ [10], yet we give a dynamic fixed-parameter algorithm for it. Hence, the existence of a polynomial kernel for a parameterized problem is not related to the existence of a dynamic fixed-parameter algorithm for it.

Preliminaries. We assume familiarity with basic combinatorial algorithms, especially graph algorithms and hitting set algorithms. When referring to a graph G, we will write $V(G)$ to denote its vertex set and $E(G)$ to denote its edge set. Unless otherwise specified, n and m will refer to the number of nodes and edges in G, respectively. We use the terms nodes and vertices interchangeably. By $\tilde{O}(f(n))$ we denote $f(n)\log^{O(1)} n$. We also assume familiarity with dynamic problems and parameterized problems.

2 Overview of the Algorithmic Techniques

Promise model and Full model. There are two different models of dynamic parameterized problems in which we design algorithms: the *promise model* and the *full model*. When solving a problem with parameter k in the promise model, there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that one is promised that throughout the sequence of updates, there always

exists a solution with parameter at most $g(k)$. Hence, one only needs to maintain a solution under updates with good guarantees on both query and update times as long as the promise continues to hold. If at any point during the execution no solution to the parameterized problem with parameter $g(k)$ exists, then the algorithm is not required to provide any guarantees.

In the full model, there is no such promise. One needs to efficiently maintain a solution with parameter at most k, or the fact that no such solution exists, under any sequence of updates. When possible, it is desirable to have an algorithm with guarantees in the full model instead of only the promise model, and all but two of our algorithms (Point Line Cover and Edge Clique Cover) do work in the full model.

2.1 Techniques for designing dynamic fixed-parameter algorithms

We present two main techniques for obtaining dynamic fixed-parameter algorithms: dynamic kernels and dynamic branching trees.

Dynamization via kernelization. Using the notation of Cygan et al. [21], a *kernelization algorithm* for a parameterized problem Π is an algorithm $\mathcal{A}$ that, given an instance (I, k) of Π, runs in polynomial time and returns an instance (I', k') of Π such that the size of the new instance is bounded by a computable function of k and so that (I', k') is a 'yes' instance of Π if and only if (I, k) is. Frequently, when the problem asks us to output more than just a Boolean answer, then an answer for (I', k') must be valid for (I, k) as well. We will refer to the output of $\mathcal{A}$ as a *kernel*. For example, a kernelization algorithm for Vertex Cover might take as input a graph G, and return a subgraph G' such that any vertex cover of G' is also a vertex cover of G.

In the first approach, we compute a kernel for the problem, and maintain that this is a valid kernel as we receive updates. In other words, as we receive updates, we will maintain what the output of a kernelization algorithm $\mathcal{A}$ would be, without actually rerunning $\mathcal{A}$ each time. Similar to kernelizations for static fixed-parameter algorithms, if we can prove that the size of our kernel is only a function of k whenever a solution with parameter k exists, then we can answer queries in time independent of n by running the fastest known static algorithms on the kernel.

The difficult part, then, is to efficiently dynamically maintain the kernel. The details of how efficiently we can handle updates to the kernel also determines which model of dynamic fixed-parameter algorithm our algorithm works for. If the kernel is defined by sufficiently simple or local rules such that updates can take place in time independent of the current kernel size, then the algorithm should work in the full model. If updates might take time linear in the kernel size, then the algorithm only works in the promise model.

As we will see, there are many problems for which we can efficiently maintain a kernel. In some instances we will be able to maintain the classical kernels known for the corresponding static problem, while in others, we will design new kernels which are easier to maintain.

Dynamization via branching tree. In the second approach, we consider so-called *set selection problems*. In these problems, the instance consists of a set of objects U (e.g., vertices of a graph), the parameter is k, and one needs to select a subset $S \subseteq U$ of size k (at least k/at most k) so that a certain predicate $P(S)$ is satisfied. Many parameterized problems are of this nature, such as k-Path, Vertex Cover, and (Directed) Feedback Vertex Set.

Consider a (static) set selection problem which admits a branching solution. By this we mean, for every instance U of the problem, there is an 'easy to find' subset $T \subseteq U$ of size

$|T| \leq f(k)$ (for some function f) so that any solution S of size at most k must intersect T. Furthermore, for any choice of $t \in T$ to be placed in the solution, one can efficiently obtain a reduced instance of the problem with parameter $k-1$, which corresponds to picking t to be in the solution. For instance, for Vertex Cover, every edge $\{u, v\}$ can be viewed as such a subset T since at least one of u and v is in any vertex cover, and if we pick u, then we can remove it and all its incident edges from the graph to get a reduced instance.

For such problems, there is a simple fixed-parameter algorithm called the branching tree algorithm: The algorithm can be represented as a tree $\mathcal{T}$ rooted at a node r. Each node v of the tree corresponds to a reduced instance of the original one, and in this instance, v has a subset T of size $f(k)$, and a child v_i for every $i \in T$, where v_i corresponds to selecting i to be placed in the solution, and v_i carries the reduced instance where i is selected. The height of the tree X is bounded by k since at most k elements need to be selected, and the branching factor is $f(k)$. Each leaf ℓ of the tree $\mathcal{T}$ is either a "yes"-leaf (when the predicate is satisfied on the set of elements selected on the path from r to ℓ) or a "no"-leaf (when the predicate is not satisfied). The runtime of the algorithm bounded by $f(k)^k \cdot t(N)$, where $t(N)$ is the time to find a subset T that must contain an element of the solution in instances of size N, together with the time to find a reduced instance, once an element is selected.

What we have described so far is a static algorithmic technique, but we investigate when this algorithmic technique can be made dynamic. In other words, given an update, we would like to quickly update $\mathcal{T}$ so that it becomes a valid branching tree for the updated instance. Since the number of nodes in the branching tree is only a function of k, one can afford to look at every tree node. Ideally, one would like the time spent per node to only depend on k. However, for most problems that we consider, the branching tree needs to be rebuilt every so often, since the subset T to branch on may become invalid after an update, and the time to rebuild can have a dependence on the instance size. We use two methods to avoid this. The first is to randomize the decisions made in the branching tree (e.g., which set T to pick) so that, assuming an oblivious adversary that must provide the update sequence in advance, it is relatively unlikely that we need to rebuild the tree $\mathcal{T}$ (or its subtrees) after each update, and in particular, so that the expected cost of an update is only a function of k. The second is to make 'robust' choices of T, so that many updates are requires before the choice of T becomes incalid, and then amortize the cost of rebuilding the tree over all the updates required to force such a rebuilding.

2.2 Algorithm Examples

We give overviews of the techniques used in some of our algorithms, to demonstrate the dynamic kernel and dynamic branching tree approaches, and different ways in which they can be used. We emphasize that these descriptions are substantial simplifications which hide many non-trivial details and ideas.

Vertex Cover. We give both a dynamic kernel algorithm and a dynamic branching tree algorithm for Vertex Cover.

Our first algorithm maintains a kernel obtained as follows: Every node of degree $\geq k+1$ 'selects' $k+1$ incident edges arbitrarily and adds them to the edge set E' of the kernel, independently of other nodes. Next, every edge incident to two nodes of degree $\leq k$ is also added to E'. Finally, the node set of the kernel consists of all nodes that are not isolated in E'. This is a valid kernel, since any vertex cover of size at most k needs to include every vertex of degree strictly greater than k. Every edge in E' either has both its end points of degree $\leq k$, or is selected by one of its end points of degree at least $k+1$. Any node of a

vertex cover of the kernel either has degree $\leq k$ or selects $k+1$ edges. Thus the kernel must have size $O(k^2)$ when a k-vertex cover exists. To insert an edge we simply add the edge to the kernel unless one of its incident vertices has degree greater than k. If one of the end points x used to be of degree $\leq k$ and is now of degree $k+1$, we have x select all its incident edges and add them to the kernel. To delete an edge, we simply remove it from the kernel. If it was incident to a vertex v of degree higher than $k+2$, then we need to find another edge incident to v which is in the graph but not selected by v to put into the kernel. If one of the end points now has degree k, we need to go through the incident edges and remove them from the kernel if their other end point has high degree and did not select them. All these operations can be performed by storing appropriate pointers so that the updates run in $O(k)$ time. With a little bit more work one can make them run in $O(1)$ amortized time. To answer queries, we answer "no" in constant time if the kernel has more than $2k(k+1)$ edges, and otherwise we run the fastest static k-Vertex Cover fixed-parameter algorithm on the kernel of size $O(k^2)$. This results in a $O(1.2738^k)$ update time.

Our second algorithm maintains a branching tree of depth at most k, which corresponds to using following randomized branching strategy: pick a uniformly random edge, and branch on adding each of its endpoints into the vertex cover. For a static branching algorithm, there is no need to pick a uniformly random edge to branch on, since at least one endpoint of every single edge must be in the vertex cover. However, a deterministic branching strategy like this in a dynamic algorithm would be susceptible to an adversarial edge update sequence, in which the adversary frequently removes edges which have been chosen to branch on. By ensuring that each edge we branch on is a uniformly random edge, we make the probability that we need to recompute any subtree of the branching tree $\mathcal{T}$ low. We compute the expected update time to be only $O(k2^k)$. Queries can be answered in only $O(k)$ time by following a path in the branching tree to an accepting leaf, if one exists.

These algorithms demonstrate some subtleties of the two techniques. In the branching tree algorithm, we use a randomized branching rule to deal with adversarial updates. In some of our other branching tree algorithms, like for Feedback Vertex Set, we are able to find a deterministic branching rule to yield a deterministic algorithm instead. In the kernelization algorithm, we manage to find a kernel which can be updated quickly even when the answer becomes larger than k and the kernel size becomes large. In other problems, it will be harder to do this, and we may need to restrict ourselves to the promise model where we are guaranteed that the kernel will not grow too big in order to have efficient update times. Dynamic kernelization techniques typically lead to faster update times and query times, like in this case, because we can apply the fastest known static algorithm for the problem to the kernel to answer queries. In a branching tree algorithm, we may be using a branching rule which does not lead to the fastest algorithm because it is easier to dynamically maintain.

Interestingly, we are able to generalize both of these algorithms to the d-Hitting Set problem. The d-Hitting Set branching tree algorithm is similar to that of Vertex Cover, but the d-Hitting Set dynamic kernelization algorithm is much more complicated, and involves a tricky recursive rule for determining which sets to put in the kernel. We include an overview of the static kernel construction in Sect. 3.

Max Leaf Spanning Tree. Our algorithm for Max Leaf Spanning Tree uses the dynamic kernel approach. The kernel we maintain is simply the given graph, where we contract vertices of degree two whose neighbors both also have degree two. We can maintain this kernel by storing paths of contracted vertices in lists corresponding to edges they have been contracted into. As long as this kernel has $\Omega(k^2)$ nodes, it must always have a spanning tree with at least k leaves.

Unlike in other dynamic kernel algorithms, where we maintain that the kernel does not get too large, this kernel may grow to have $\Omega(n^2)$ edges. We can nonetheless find a tree with at least k leaves in time independent of n, by breadth-first searching from an arbitrary node in the kernel until we have $\Omega(k^2)$ kernel vertices, and just finding a tree within those vertices.

This method finds a subtree T_S with at least k leaves, but we need to find a tree which spans the whole graph. In the static problem, this could be accomplished by a linear-time breadth-first search away from T_S, but in the dynamic problem, this is too slow. To overcome this, we also maintain a spanning tree T of the entire graph, which does not necessarily have k leaves, using a known dynamic tree data structure. When queried for a spanning tree, we find T_S, and then perform a 'merge' operation to combine T and T_S into a spanning tree with at least k leaves. This merge operation makes only $O(k^4)$ changes to T, so we are able to maintain a desired spanning tree in time independent of n.

We are able to maintain a linear size answer in only logarithmic time per update because the output is not very 'sensitive' to updates: we can always output an answer very close to T, which itself only changes in one edge per update. In other problems where the output can be more sensitive to updates, like Edge Clique Cover, we need to maintain a small intermediate representation of the answer instead of the answer itself.

Feedback Vertex Set in undirected graphs. Our algorithm for Feedback Vertex Set combines the dynamic kernel approach with the dynamic branching tree approach. We will maintain a branching tree, where we branch off of which node to include in our feedback vertex set. Then, at each node in the branching tree, we will maintain a kernel to help decide what nodes to branch on. Similar to the situation with Max Leaf Spanning Tree, our kernel can possibly have $\Omega(n^2)$ edges. Here we will deal with this by branching off of only $O(k)$ nodes in the kernel to add to our feedback vertex set, so that we can answer queries in sublinear time in the kernel size.

The kernel we maintain at each node of the branching tree is the given graph, in which vertices of degree one are deleted, and vertices of degree two are contracted. This involves many details for maintaining contracted trees, and dealing with resulting self-loops. Since the resulting graph has average degree at least three, whereas forests have much lower average degree, we show that a feedback vertex set of size at most k must contain a vertex of high degree, whose degree is at least $1/(3k)$ of the total number of edges in the kernel. Since there are at most $6k$ such vertices, we can branch on which to include in our feedback vertex set.

This branching strategy works well for the static problem, but it is hard to maintain dynamically. Each edge update might change the set of vertices with high enough degree to branch on, and changing which vertex we branch on, and recomputing an entire subtree of the branching tree, can be expensive. We alleviate this issue using amortization. Instead of branching only on the $6k$ highest degree vertices, we instead branch on the $12k$ highest degree vertices. If our kernel has m edges, then we prove that $\Omega(m)$ edge updates need to happen before there might be a small feedback vertex set containing none of the vertices we branched on. After these updates we need to recompute the branching tree, but this is inexpensive when amortized over the required $\Omega(m)$ updates.

3 A dynamic kernel for d-Hitting Set

In this section we present our dynamic kernel for the d-Hitting Set problem; we describe how to efficiently compute and maintain it in the full version of the paper. The d-Hitting Set problem asks to find a set $X \subseteq U$ of at most k elements of a given a universe U which

intersects all sets from a family $\mathcal{F}$ of subsets of U, each of cardinality exactly d. Here we present a kernel for d-Hitting Set, with $(d-1)!k(k+1)^{d-1}$ sets and $d!k(k+1)^{d-1}$ elements. It is known [25] that a kernel of size $O(k^{d-\varepsilon})$ for any constant $\varepsilon > 0$ would imply that $\mathsf{NP} \subseteq \mathsf{coNP}/poly$; thus, the k^d dependence on the number of sets in the kernel is optimal.

Let us describe the kernel. We will recursively define the notion of a *good* set.

▶ **Definition 4** (good set)**.** Let $r \in \mathbb{N}$ and $\nu_r = r!(k+1)^r$. Let $d \in \mathbb{N}$ and let $(U, \mathcal{F})$ be an instance of d-Hitting Set. We define the notion of an "(ℓ, r)-good" set inductively, in decreasing order of ℓ from d to 1, and for fixed ℓ, for increasing r from 1 to $d - \ell$.

- Any set $S \in \mathcal{F}$ is (d, r)-*good* for all r.
- A set $S \subseteq U$ is ℓ-*good* if S is (ℓ, r)-good for some r.
- A set $S \subseteq U$ is (ℓ', r)-*strong* if S is ℓ'-good and does not contain any $(\ell' - j, j)$-good subsets for any $j \in \{1, \ldots, r-1\}$.
- A set $S \subseteq U$ is (ℓ, r)-*good* if $|S| = \ell$ and S is a subset of at least ν_r $(\ell + r, r)$-strong sets.
- A set $S \subseteq U$ is *good* if S is ℓ-good for $\ell = |S|$.

Notice that if a set is (ℓ', r)-strong, then it is also (ℓ', r')-strong for all $r' < r$. Also, any ℓ-good set is $(\ell, 1)$-strong. Further, note that since the notion of $(\ell + r, r)$-strong only depends on $(\ell + a, r - a)$-good sets for $a \geq 1$, the definition of (ℓ, r)-good is sound.

Let $\mathcal{F}'$ consist of those $S \subseteq U$ that are good and none of their subsets are good. Let U' consist of all $u \in U$ that are contained in some set of $\mathcal{F}'$. Let $K = (U', \mathcal{F}')$. In the full version we prove the following lemma that shows that (K, k) is a kernel for the instance $(U, \mathcal{F})$.

▶ **Lemma 5.** *Let $(U, \mathcal{F})$ be an instance of d-Hitting Set. If $(U, \mathcal{F})$ admits a hitting set X of size at most k, then any good set $S \subseteq U$ intersects X non-trivially.*

The lemma implies that (K, k) is a kernel: first, if X' is a hitting set of K, it is a hitting set for $\mathcal{F}$ as well since for every $F \in \mathcal{F}$, either $F \in \mathcal{F}'$ or some subset of F is in $\mathcal{F}'$. Now let X be a hitting set of $\mathcal{F}$ with size at most k. By the lemma, if some S is in $\mathcal{F}'$, then it intersects X non-trivially and so X is a hitting set of $\mathcal{F}'$ as well. Now we argue about the size of K.

▶ **Lemma 6.** *If $(U, \mathcal{F})$ (and hence also $(U', \mathcal{F}')$) admits a hitting set of size at most k, then $|U'| \leq d|\mathcal{F}'|$ and $|\mathcal{F}'| \leq \left(1 + \frac{2}{(k+1)(d-1)}\right) \cdot d!(k+1)^d$.*

Proof. If $\{u\} \in \mathcal{F}'$, then no other set containing u can be in $\mathcal{F}'$. Otherwise, consider some u such that $\{u\} \notin \mathcal{F}'$. Consider all sets of size $r+1$ in $\mathcal{F}'$ that contain u, for any choice of $r \in \{1, \ldots, d-1\}$.

Since $\{u\} \notin \mathcal{F}'$, we know that u cannot be $(1, r)$-good, and thus u is contained in fewer than ν_r $(r+1)$-good sets that do not contain any $(j+1, r-j)$-good subsets for any $j \in \{2, \ldots, r-1\}$. Now since for every $F \in \mathcal{F}'$ we have that it contains no good subsets, this means that u is contained in fewer than ν_r sets in $\mathcal{F}'$ of size $r+1$.

Thus, the number of sets of $\mathcal{F}'$ containing u is at most

$$\sum_{r=1}^{d-1} \nu_r = \sum_{r=1}^{d-1} r!(k+1)^r \leq \left(1 + \frac{2}{(k+1)(d-1)}\right)(d-1)!(k+1)^{d-1},$$

where the last inequality can be proven inductively. Thus, if there is a hitting set of size at most k for $\mathcal{F}'$, then the size of $\mathcal{F}'$ is at most $(1 + \frac{2}{(k+1)(d-1)})d!(k+1)^d$. ◀

In the full version we additionally show that the kernel can be computed in $O(3^d n + m)$ time and can be dynamically maintained with very fast updates.

Acknowledgments. The authors would like to thank Nicole Wein, Daniel Stubbs, Hubert Teo, and Ryan Williams for fruitful conversations.

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. FOCS 2014*, pages 434–443, 2014.

2 Faisal N. Abu-Khzam, Judith Egan, Michael R. Fellows, Frances A. Rosamond, and Peter Shaw. On the parameterized complexity of dynamic problems with connectivity constraints. In *Proc. COCOA 2014*, volume 8881 of *Lecture Notes Comput. Sci.*, pages 625–636, 2014.

3 Faisal N. Abu-Khzam, Judith Egan, Michael R. Fellows, Frances A. Rosamond, and Peter Shaw. On the parameterized complexity of dynamic problems. *Theor. Comput. Sci.*, 607:426–434, 2015.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4), 1995.

5 Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully dynamic maximal matching in $O(\log n)$ update time. *SIAM J. Comput.*, 44(1):88–113, 2015.

6 Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *J. Artif. Intelligence Res.*, 12:219–234, 2000.

7 Aaron Bernstein and Liam Roditty. Improved dynamic algorithms for maintaining approximate shortest paths under deletions. In *Proc. SODA 2011*, pages 1355–1365, 2011.

8 Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. In *Proc. SODA 2015*, pages 785–804, 2015.

9 Hans L. Bodlaender. Dynamic algorithms for graphs with treewidth 2. In *Proc. WG 1993*, volume 790 of *Lecture Notes Comput. Sci.*, pages 112–124, 1993.

10 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

11 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

12 S. Buss. private communication.

13 Leizhen Cai, Siu Man Chan, and Siu On Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proc. IPEC 2006*, volume 4169 of *Lecture Notes Comput. Sci.*, pages 239–250, 2006.

14 Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Ann. Pure Applied Logic*, 84(1):119–138, 1997.

15 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoret. Comput. Sci.*, 411(40):3736–3756, 2010.

16 Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5), 2008.

17 Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proc. SODA 2015*, pages 1234–1251, 2015.

18 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

19 Marek Cygan. Deterministic parameterized connected vertex cover. In *Proc. SWAT 2012*, volume 7357 of *Lecture Notes Comput. Sci.*, pages 95–106, 2012.

20 Marek Cygan, Fedor Fomin, Bart M.P. Jansen, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Open problems for FPT school 2014. `http://fptschool.mimuw.edu.pl/opl.pdf`.

21 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, Cham, 2015.

22 Marek Cygan, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. Clique cover and graph separation: New incompressibility results. *ACM Trans. Comput. Theory*, 6(2):6:1–6:19, 2014.

23 Marek Cygan, Marcin Pilipczuk, and Michał Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM J. Comput.*, 45(1):67–83, 2016.

24 Jean Daligault, Gregory Gutin, Eun Jung Kim, and Anders Yeo. FPT algorithms and kernels for the directed k-leaf problem. *J. Comput. Syst. Sci.*, 76(2):144–152, 2010.

25 Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014.

26 Camil Demetrescu and Giuseppe F. Italiano. A new approach to dynamic all pairs shortest paths. *J. ACM*, 51(6):968–992, 2004.

27 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In *Proc. ICALP 2009*, volume 5555 of *Lecture Notes Comput. Sci.*, pages 378–389, 2009.

28 Frederic Dorn. Planar subgraph isomorphism revisited. In *Proc. STACS 2010*, volume 5 of *Leibniz Int. Proc. Informatics*, pages 263–274, 2010.

29 Zdeněk Dvořák, Martin Kupec, and Vojtěch Tůma. A dynamic data structure for MSO properties in graphs with bounded tree-depth. In *Proc. ESA 2014*, volume 8737 of *Lecture Notes Comput. Sci.*, pages 334–345, 2014.

30 Zdeněk Dvořák and Vojtěch Tůma. A dynamic data structure for counting subgraphs in sparse graphs. In *Proc. WADS 2013*, volume 8037 of *Lecture Notes Comput. Sci.*, pages 304–315, 2013.

31 Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, and Frances A. Rosamond. FPT is P-time extremal structure I. In *Proc. ACiD 2005*, pages 1–41, 2005.

32 Michael Etscheid and Matthias Mnich. Linear kernels and linear time algorithms for finding large cuts. In *Proc. ISAAC 2016*, volume 64 of *Leibniz Int. Proc. Informatics*, pages 31:1–31:13, 2016.

33 Stefan Fafianie and Stefan Kratsch. A shortcut to (sun)flowers: Kernels in logarithmic space or linear time. In *Proc. MFCS 2015*, volume 9235 of *Lecture Notes Comput. Sci.*, pages 299–310, 2015.

34 Henning Fernau. Edge dominating set: Efficient enumeration-based exact algorithms. In *Proc. IPEC 2006*, volume 4169 of *Lecture Notes Comput. Sci.*, pages 142–153, 2006.

35 Anka Gajentaan and Mark H. Overmars. On a class of problems in computational geometry. *Comput. Geom.*, 45(4):140–152, 2012.

36 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. ISSAC 2014*, pages 296–303, 2014.

37 David Gibb, Bruce Kapron, Valerie King, and Nolan Thorn. Dynamic graph connectivity with improved worst case update time and sublinear space, 2015. URL: `https://arxiv.org/abs/1509.06464`.

38 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *J. Exp. Algorithmics*, 13:2:2.2–2:2.15, 2009.

39 Manoj Gupta and Richard Peng. Fully dynamic $(1+\varepsilon)$-approximate matchings. In *Proc. FOCS 2013*, pages 548–557, 2013.

40 Andras Gyárfás. A simple lower bound on edge coverings by cliques. *Discrete Math.*, 85(1):103–104, 1990.

41 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Improved algorithms for decremental single-source reachability on directed graphs. In *Proc. ICALP 2015*, volume 9134 of *Lecture Notes Comput. Sci.*, pages 725–736, 2015.

42 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Dynamic approximate all-pairs shortest paths: Breaking the $O(mn)$ barrier and derandomization. *SIAM J. Comput.*, 45(3):947–1006, 2016.

43 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proc. FOCS 2015*, pages 21–30, 2015.

44 Monika Rauch Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, 1999.

45 Monika Rauch Henzinger and Valerie King. Maintaining minimum spanning forests in dynamic graphs. *SIAM J. Comput.*, 31(2):364–374, 2001.

46 Monika Rauch Henzinger and Mikkel Thorup. Sampling to provide or to bound: With applications to fully dynamic graph algorithms. *Random Struct. Algorithms*, 11(4):369–379, 1997.

47 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

48 Shang-En Huang, Dawei Huang, Tsvi Kopelowitz, and Seth Pettie. Fully dynamic connectivity in $O(\log n(\log\log n)^2)$ amortized expected time, 2016. URL: `http://arxiv.org/abs/1609.05867`.

49 Ken Iwaide and Hiroshi Nagamochi. An improved algorithm for parameterized edge dominating set problem. *J. Graph Algorithms Appl.*, 20(1):23–58, 2016.

50 Yoichi Iwata. A linear time kernelization for feedback vertex set, 2016. URL: `https://arxiv.org/abs/1608.01463`.

51 Yoichi Iwata and Keigo Oka. Fast dynamic graph algorithms for parameterized problems. In *Proc. SWAT 2014*, volume 8503 of *Lecture Notes Comput. Sci.*, pages 241–252, 2014.

52 Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time FPT algorithms via network flow. In *Proc. SODA 2014*, pages 1749–1761, 2014.

53 Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proc. SODA 2013*, pages 1131–1142, 2013.

54 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proc. SODA 2016*, pages 1272–1287, 2016.

55 Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point line cover: The easy kernel is essentially tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016.

56 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *Proc. FOCS 2012*, pages 450–459, 2012.

57 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014.

58 Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. Linear time parameterized algorithms for subset feedback vertex set. In *Proc. ICALP 2015*, volume 9134 of *Lecture Notes Comput. Sci.*, pages 935–946, 2015.

59 Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. A linear time parameterized algorithm for directed feedback vertex set, 2016. URL: `https://arxiv.org/abs/1609.04347`.

60 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Proc.FOCS 2013*, pages 253–262, 2013.

61 Krzysztof Onak and Ronitt Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proc. STOC 2010*, pages 457–464, 2010.

62 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proc. STOC 2010*, pages 603–610, 2010.

63 Mihai Pătraşcu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011.

64 Mihai Pătraşcu and Erik D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput.*, 35(4):932–963, 2006.

65 Mihai Pătraşcu and Mikkel Thorup. Don't rush into a union: take time to find your roots. In *Proc. STOC 2011*, pages 559–568, 2011.

66 Shay Solomon. Fully dynamic maximal matching in constant update time. In *Proc. FOCS 2016*, pages 325–334, 2016.

67 Mikkel Thorup. Near-optimal fully-dynamic graph connectivity. In *Proc. STOC 2000*, pages 343–350, 2000.

68 René van Bevern. Towards optimal and expressive kernelization for d-hitting set. *Algorithmica*, 70(1):129–147, 2014.

69 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. STOC 2012*, pages 887–898, 2012.

70 Magnus Wahlström. Half-integrality, LP-branching and FPT algorithms. In *Proc. SODA 2014*, pages 1762–1781, 2014.

71 Christian Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *Proc. SODA 2013*, pages 1757–1769, 2013.

Decremental Data Structures for Connectivity and Dominators in Directed Graphs*†

Loukas Georgiadis[1], Thomas Dueholm Hansen[2], Giuseppe F. Italiano[3], Sebastian Krinninger[4], and Nikos Parotsidis[5]

1 **University of Ioannina, Ioannina, Greece**
loukas@cs.uoi.gr
2 **Aarhus University, Aarhus, Denmark**
tdh@cs.au.dk
3 **University of Rome Tor Vergata, Rome, Italy**
giuseppe.italiano@uniroma2.it
4 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
sebastian.krinninger@univie.ac.at
5 **University of Rome Tor Vergata, Rome, Italy**
nikos.parotsidis@uniroma2.it

Abstract

We introduce a new dynamic data structure for maintaining the strongly connected components (SCCs) of a directed graph (digraph) under edge deletions, so as to answer a rich repertoire of connectivity queries. Our main technical contribution is a decremental data structure that supports sensitivity queries of the form "are u and v strongly connected in the graph $G \setminus w$?", for any triple of vertices u, v, w, while G undergoes deletions of edges. Our data structure processes a sequence of edge deletions in a digraph with n vertices in $O(mn \log n)$ total time and $O(n^2 \log n)$ space, where m is the number of edges before any deletion, and answers the above queries in constant time. We can leverage our data structure to obtain decremental data structures for many more types of queries within the same time and space complexity. For instance for edge-related queries, such as testing whether two query vertices u and v are strongly connected in $G \setminus e$, for some query edge e.

As another important application of our decremental data structure, we provide the first nontrivial algorithm for maintaining the dominator tree of a flow graph under edge deletions. We present an algorithm that processes a sequence of edge deletions in a flow graph in $O(mn \log n)$ total time and $O(n^2 \log n)$ space. For reducible flow graphs we provide an $O(mn)$-time and $O(m + n)$-space algorithm. We give a conditional lower bound that provides evidence that these running times may be tight up to subpolynomial factors.

1998 ACM Subject Classification E.1 [Graphs and Networks] Trees, F.2.2 Computations on Discrete Structures, G.2.2 [Graph Algorithms] Trees

Keywords and phrases dynamic graph algorithms, decremental algorithms, dominator tree, strong connectivity under failures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.42

* Full version of this paper available at https://arxiv.org/abs/1704.08235.

† The work of L. Georgiadis and T. D. Hansen was partially done while visiting University of Rome Tor Vergata. T. D. Hansen was supported by the Carlsberg Foundation, grant no. CF14-0617. G. F. Italiano was partially supported by the Italian Ministry of Education, University and Research, under Project AMANDA. The work of S. Krinninger was partially done while visiting University of Rome Tor Vergata and while at Max Planck Institute for Informatics, Saarland Informatics Campus, Germany.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 42; pp. 42:1–42:15

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

1 Introduction

Dynamic graph algorithms have been extensively studied for several decades, and many important results have been achieved for dynamic versions of fundamental problems, including connectivity, 2-edge and 2-vertex connectivity, minimum spanning tree, transitive closure, and shortest paths (see, e.g., the survey in [14]). We recall that a dynamic graph problem is said to be *fully dynamic* if it involves both insertions and deletions of edges, *incremental* if it only involves edge insertions, and *decremental* if it only involves edge deletions.

The decremental strongly connected components (SCCs) problem asks us to maintain, under edge deletions in a directed graph G, a data structure that given two vertices u and v answers whether u and v are strongly connected in G. We extend this problem to sensitvity queries of the form "are u and v strongly connected in the graph $G \setminus w$?", for any triple of vertices u, v, w, i.e., we additionally allow the query to temporarily remove a third vertex w. We show that this extended decremental SCC problem can be used to answer fast a rich repertoire of connectivity queries, and we present a new and efficient data structure for the problem. In particular, our data structure for the extended decremental SCC problem can be used to support edge-related queries, such as maintaining the strong bridges of a digraph, testing whether two query vertices u and v are strongly connected in $G \setminus e$, reporting the SCCs of $G \setminus e$, or the largest and smallest SCCs in $G \setminus e$, for any query edge e. Furthermore, using our framework, it is possible to maintain the 2-vertex-and 2-edge-connected components of a digraph under edge deletions. All of these extensions can be handled with the same time and space bounds as for the extended decremental SCC problem. (Most of these reductions have been deferred to the full version of the paper.)

A naive approach to solving the extended decremental SCC problem is to maintain separately the SCCs in every subgraph $G \setminus w$ of G, for all vertices w. After an edge deletion we then update the SCCs of all these n subgraphs, where n is the number of vertices in G. If we simply perform a static recompution after each deletion, then we, for example, obtain decremental algorithms with $O(m^2 n)$ total time and $O(n^2)$ space by recomputing the SCCs in each $G \setminus w$ [37] or $O(m^2 + mn)$ total time and $O(m + n)$ space by constructing a more suitable static connectivity data structure [22], respectively. Here m denotes the initial number of edges. The current fastest (randomized) decremental SCC algorithm by Chechik et al. [10] trivially gives $O(mn^{3/2} \log n)$ total update time and $O(mn)$ space for our extended decremental SCC problem.

The main technical contribution of this paper is a data structure for the extended decremental SCC problem with $O(mn \log n)$ total update time that uses $O(n^2 \log n)$ space, and that answers queries in constant time. We obtain this data structure by extending Łącki's decremental SCC algorithm [30]. His algorithm maintains the SCCs of a graph under edge deletions by recursively decomposing the SCCs into smaller and smaller subgraphs. We therefore refer to his data structure as an SCC-decomposition. His total update time is $O(mn)$ and the space used is $O(m + n)$. We observe that the naive algorithm based on SCC-decompositions can be implemented in such a way that most of the work performed is redundant. We obtain our data structure by merging n SCC-decompositions into one joint data structure, which we refer to as a joint SCC-decomposition. Our data structure, like that of Łącki, is deterministic. Using completely different techniques, Georgiadis et al. [21] showed how to answer the same sensitivity queries in $O(mn)$ total time in the incremental setting, i.e., when the input digraph undergoes edge insertions only.

The extended SCC problem is related to the so-called *fault-tolerant model*. Here, one wishes to preprocess a graph G into a data structure that is able to answer fast certain

sensitivity queries, i.e., given a failed vertex w (resp., failed edge e), compute a specific property of the subgraph $G \setminus w$ (resp., $G \setminus e$) of G. Our data structure supports sensitivity queries when a digraph G undergoes edge deletions, which gives an aspect of *decremental fault-tolerance*. This may be useful in scenarios where we wish to find the best edge whose deletion optimizes certain properties (fault-tolerant aspect) and then actually perform this deletion (decremental aspect). This is, e.g., done in the computational biology applications considered by Mihalák et al. [32]. Their recursive deletion-contraction algorithm repeatedly finds the edge of a strongly connected digraph whose deletion maximizes quantities such as the number of resulting SCCs or minimizes their maximum size.

As another important application of our joint SCCs data structure, we provide the first nontrivial algorithm for maintaining the dominator tree of a flow graph under edge deletions. A flow graph $G = (V, E, s)$ is a directed graph with a distinguished start vertex $s \in V$, w.l.o.g. containing only vertices reachable from s. A vertex w *dominates* a vertex v (w is a *dominator* of v) if every path from s to v includes w. The *immediate dominator* of a vertex v, denoted by $d(v)$, is the unique vertex that dominates v and is dominated by all dominators of v. The *dominator tree* D is a tree with root s in which each vertex v has $d(v)$ as its parent. Dominator trees can be computed in linear time [2, 7, 8, 23]. The problem of finding dominators has been extensively studied, as it occurs in several applications, including program optimization and code generation [12], constraint programming [34], circuit testing [4], theoretical biology [1], memory profiling [31], fault-tolerant computing [5, 6], connectivity and path-determination problems [16, 17, 19, 18, 25, 27, 28, 29], and the analysis of diffusion networks [24].

In particular, the dynamic dominator problem arises in various applications, such as data flow analysis and compilation [11, 15]. Moreover, the results of Italiano et al. [27] imply that dynamic dominators can be used for dynamically testing 2-vertex connectivity, and for maintaining the strong bridges and strong articulation points of digraphs. The decremental dominator problem appears in the computation of maximal 2-connected subgraphs in digraphs [25, 29, 13]. The problem of updating the dominator relation has been studied for a few decades (see, e.g., [3, 9, 11, 20, 33, 35, 36]). For the incremental dominator problem, there are algorithms that achieve total $O(mn)$ running time for processing a sequence of edge insertion in a flow graph with n vertices, where m is the number of edges after all insertions [3, 11, 20]. Moreover, they can answer dominance queries, i.e., whether a query vertex w dominates another query vertex v, in constant time. Prior to our work, to the best of our knowledge, no decremental algorithm with total running time better than $O(m^2)$ was known for general flow graphs. In the special case of *reducible* flow graphs (a class that includes acyclic flow graphs), Cicerone et al. [11] achieved an $O(mn)$ update bound for the decremental dominator problem. Both the incremental and the decremental algorithms of [11] require $O(n^2)$ space, as they maintain the transitive closure of the digraph.

Our algorithm is the first to improve the trivial $O(m^2)$ bound for the decremental dominator problem in *general* flow graphs. Specifically, our algorithm can process a sequence of edge deletions in a flow graph with n vertices and initially m edges in $O(mn \log n)$ time and $O(n^2 \log n)$ space, and after processing each deletion can answer dominance queries in constant time. For the special case of *reducible* flow graphs, we give an algorithm that matches the $O(mn)$ running time of Cicerone et al. while improving the space usage to $O(m+n)$. We remark that the reducible case is interesting for applications in program optimization since one notion of a "structured" program is that its flow graph is reducible. (The details about this result appear in the full paper.) Finally, we complement our results with a conditional lower bound, which suggests that it will be hard to substantially improve our update bounds. In particular, we prove that there is no incremental nor decremental algorithm for maintaining

the dominator tree (or more generally, a dominance data structure) that has total update time $O((mn)^{1-\epsilon})$ (for some constant $\epsilon > 0$) unless the OMv Conjecture [26] fails. The same lower bound applies to the extended decremental SCC problem. Unlike the update time, it is not clear that the $O(n^2 \log n)$ space used by our joint SCC-decomposition is near-optimal. We leave it as an open problem to improve this bound.

Further Notation and Terminology. For a given digraph $G = (V, E)$, we denote the set of vertices by $V(G) = V$ and the set of edges by $E(G) = E$. We let $|E| = m$ and $|V| = n$. Two vertices u and v are *strongly connected* in G if there is a path from u to v and a path from v to u. G is *strongly connected* if every vertex is reachable from every other vertex. The *strongly connected components* (SCCs) of G are its maximal strongly connected subgraphs. We denote by $G \setminus S$ (resp., $G \setminus e$)) the graph obtained after deleting a set S of vertices (resp., an edge e) from G. For a strongly connected graph H, we say that deleting an edge e *breaks* H, if $H \setminus e$ is not strongly connected.

An edge (resp., a vertex) of G is a *strong bridge* (resp., a *strong articulation point*) if its removal increases the number of SCCs. An edge e (resp., a vertex v) is a *separating edge* (resp., a *separating vertex*) for two vertices u and v if u and v are not strongly connected in $G \setminus e$ (resp., in $G \setminus v$). Two vertices u and v are 2-*edge connected* (2-*vertex connected*) if there are two edge-disjoint paths (internally vertex-disjoint paths) between u and v. We denote by G^{R} the *reverse graph* of G, i.e., the graph which has the same vertices as G and contains an edge $e^{\mathrm{R}} = (v, u)$ for every edge $e = (u, v)$ of G.

2 A Data Structure for Maintaining Joint SCC-Decompositions

For a given initial graph G, the *decremental SCC problem* asks us to maintain a data structure that allows edge deletions and can answer whether (arbitrary) pairs (u, v) of vertices are in the same SCC. The goal is to update the data structure as quickly as possible while answering queries in constant time. In this paper we present a data structure for the *extended decremental SCC problem* in which a query provides an additional vertex w and asks whether u and v are in the same SCC when w is deleted from G. We maintain this information under edge deletions, and our data structure relies on Łącki's *SCC-decomposition* [30] for doing so.

2.1 Review of Łącki's SCC Decomposition

An SCC-decomposition recursively partitions the graph G into smaller strongly connected subgraphs. This generates a rooted tree T, whose root r represents the entire graph, and where the subtree rooted at each node ϕ represents some vertex-induced strongly connected subgraph G_ϕ (we refer to vertices of T as nodes to distinguish T from G). Every non-leaf node ϕ is a vertex of G_ϕ, and the children of ϕ correspond to SCCs of $G_\phi \setminus \phi$. The concept was introduced by Łącki [30] and slightly extended by Chechik et al. [10]. We adopt the notation from [10].

▶ **Definition 1** (SCC-decomposition)**.** Let $G = (V, E)$ be a strongly connected graph. An *SCC-decomposition* of G is a rooted tree T, whose nodes form a partition of V. For a node ϕ of T we define G_ϕ to be the subgraph of G induced by the union of all descendants of ϕ (including ϕ). Then, the following properties hold:

- Each internal node ϕ of T is a single-element set. (In this case, we sometimes abuse notation and assume that ϕ is the vertex itself.)

- Let ϕ be any internal node of T, and let $H_1, \ldots, H_t$ be the SCCs of $G_\phi \setminus \phi$. Then the node ϕ has t children $\phi_1, \ldots, \phi_t$, where $G_{\phi_i} = H_i$ for all $i \in \{1, \ldots, t\}$.

An SCC-decomposition of a graph G that is not strongly connected is a collection of SCC-decompositions of the SCCs of G. We say that T is a *partial* SCC-decomposition when the leaves of T are not required to be singletons.

Observe that for each node ϕ, the graph G_ϕ is strongly connected. Moreover, the subtree of T rooted at ϕ is an SCC-decomposition of G_ϕ. Also, for a leaf ϕ we have that $\phi = V(G_\phi)$. To build an SCC-decomposition T of a strongly connected graph G we pick an arbitrary vertex v, put it in the root of T, then recursively build SCC-decompositions of SCCs of $G \setminus \{v\}$ and make them the children of v in T. Note that since the choice of v is arbitrary, there are many ways to build an SCC-decomposition of the same graph. Łącki [30] (and Chechik et al. [10]) introduced a procedure Build-SCC-Decomposition(G, S) that takes as input a set of vertices S and returns a partial SCC-decomposition whose internal nodes are the vertices of S, i.e., these vertices are picked first and therefore appear at the top of the constructed tree. We refer to the vertices in S as internal nodes and the remaining nodes as external nodes. Note that all external nodes appear in the leaves of T, while internal nodes can be both leaves and non-leaves. This distinction is helpful when describing our algorithm.

Łącki [30] showed that the total initialization and update time under edge deletions of an SCC-decomposition is $O(m\gamma)$, where γ is the depth of the decomposition.

2.2 Towards a Joint SCC-Decomposition

Recall that the extended decremental SCC problem asks us to maintain under edge deletions a data structure for a graph G such that we can answer whether u and v are strongly connected in $G \setminus \{w\}$ when given $u, v, w \in V(G)$. A naive algorithm does this by maintaining n SCC-decompositions, each with a distinct vertex w as its root. The children of w in an SCC-decomposition that has w as its root are then exactly the SCCs of $G \setminus \{w\}$. Hence, u and v are in the same SCC if and only if they appear in the same subtree below w. The total update time of this data structure is however $O(mn^2)$, which is undesirable. With a more refined approach, we improve the time bound to $O(mn \log n)$.

Observe that the external nodes of a partial SCC-decomposition T produced by the procedure Build-SCC-Decomposition(G, S) exactly correspond to the SCCs of $G \setminus S$. This is true regardless of the order in which vertices from S are picked by the procedure. If two SCC-decompositions are built using the same set S, but with vertices being picked in a different order, then the nodes below S represent the same SCCs, which means that they can be shared by the two SCC-decompositions. Our algorithm is based on this observation. We essentially construct the n SCC-decompositions of the naive algorithm described above such that large parts of their subtrees are shared, and such that we do not need to maintain multiple copies of these subtrees. The idea is to partition the set S into two subsets S_1 and S_2 of equal size (we assume for simplicity that n is a power of 2), and then construct half of the SCC-decompositions with S_1 at the top and the other half with S_2 at the top. The procedure is repeated recursively on the top part of both halves. We refer to the bottom part, i.e., nodes that are not from S_1 and S_2, respectively, as the *extension* of the top part. Note that we eventually get a distinct vertex as the root of each of the n SCC-decompositions. The following definition formalizes the idea.

▶ **Definition 2** (Joint SCC-decomposition). A *joint SCC-decomposition* J is a recursive structure. It is either a regular SCC-decomposition T (the base case), or a pair of joint SCC-decompositions J_1, J_2 with the same set of internal nodes S and a shared set of external nodes

Input: A graph G and a set of vertices $S \subseteq V(G)$
Output: A balanced joint SCC-decomposition $J = (J_1, J_2, S, \Phi)$ of G on S.

if $|S| = 1$ **then**
  return $T =$ Build-SCC-Decomposition(G, S).
end
Let S_1 and S_2 be the first and second half of S, respectively, and let Φ be an empty list.
foreach $i \in \{1, 2\}$ **do**
  Compute $J_i =$ Build-Joint-SCC-Decomposition(G, S_i).
  foreach *external node* ϕ *of* J_i **do**
    Compute $T_\phi =$ Build-SCC-Decomposition$(G_\phi, \phi \cap S)$.
    Add each external node of T_ϕ to Φ, if it is not already there.
  end
end
return $J = (J_1, J_2, S, \Phi)$

Figure 1 Build-Joint-SCC-Decomposition(G,S).

Φ. In the second case we refer to J as the tuple (J_1, J_2, S, Φ). A joint SCC-decomposition $J = (J_1, J_2, S, \Phi)$ is *balanced* on S if it has one of the following two properties:

1. S is a singleton and J is a regular (partial) SCC-decomposition T with the vertex from S as root and no other internal nodes (the base case).
2. S can be partitioned into two equally sized halves S_1 and S_2, and J consists of two joint SCC-decompositions $J_1 = (J_{1,1}, J_{1,2}, S_1, \Phi_1)$ and $J_2 = (J_{2,1}, J_{2,2}, S_2, \Phi_2)$ that are balanced on S_1 and S_2, respectively. Also, each external node ϕ in Φ_1 and Φ_2 is extended with an associated SCC-decomposition T_ϕ for G_ϕ whose internal nodes are those of $\phi \cap S$. The combined set of external nodes of T_ϕ for all $\phi \in \Phi_1$ is equal to the combined set of external nodes of $T_{\phi'}$ for all $\phi' \in \Phi_2$, and these nodes are the external nodes Φ of J.

The procedure Build-Joint-SCC-Decomposition(G, S) describes how we build a balanced joint SCC-decomposition. G is the graph that we wish to decompose, and S is the set of vertices that we wish to place at the top. Initially S is the set of all vertices. The following lemma bounds the number of SCC-decompositions that make up a joint balanced SCC-decomposition. The lemma is proved by observing that the number of SCC-decompositions constructed by Build-Joint-SCC-Decomposition(G, S) is given by the recurrence $g(s) = 2g(s/2) + 2$ when $s > 1$ and $g(s) = 1$ otherwise, where $s = |S|$.

▶ **Lemma 3.** *A balanced joint SCC-decomposition for a graph G with n vertices consists of $O(n)$ SCC-decompositions.*

▶ **Lemma 4.** *Let $J = (J_1, J_2, S, \Phi)$ be a balanced joint SCC-decomposition of a graph G such that $S = V(G)$. Then the total number of nodes of J is $O(n \log n)$, where $n = |V(G)|$.*

Proof. The proof is by induction. Our induction hypothesis says that the total number of internal nodes of a balanced joint SCC-decomposition $J = (J_1, J_2, S, \Phi)$, counting not only S but also recursively the number of internal nodes of J_1 and J_2, is $|S| \cdot (1 + \log |S|)$.

In the base case, J is an SCC-decomposition with a single internal node, and the induction hypothesis is clearly satisfied. For the induction step we count separately the total number of internal nodes of $J_1 = (J_{1,1}, J_{1,2}, S_1, \Phi_1)$ and $J_2 = (J_{2,1}, J_{2,2}, S_2, \Phi_2)$, and add the number of internal nodes of the SCC-decompositions T_ϕ for $\phi \in \Phi_1$ and $\phi \in \Phi_2$, i.e., the extensions

of J_1 and J_2 to S. Since $|S_1| = |S_2| = |S|/2$, it follows from the induction hypothesis that both J_1 and J_2 have $\frac{|S|}{2} \cdot \log |S|$ internal nodes in total. The internal nodes of T_ϕ for $\phi \in \Phi_1$ are exactly S_2, and the internal nodes of T_ϕ for $\phi \in \Phi_2$ are exactly S_1. Hence the number of internal nodes in the extensions are $|S_1| + |S_2| = |S|$. It follows that the total number of internal nodes of J is $|S| \cdot (1 + \log |S|)$ as desired.

It remains to count the external nodes of J. Note that external nodes of J_1 and J_2 correspond to internal nodes of J, i.e., they are roots of the SCC-decompositions that extend J_1 and J_2. Therefore there are at most as many external nodes inside the recursion as there are internal nodes in total. There are $O(n)$ external nodes in the extensions of J_1 and J_2 to S, and we conclude that the total number of nodes when $S = V(G)$ is at most $O(n \log n)$. ◀

Recall that an SCC-decomposition of depth γ can be initialized in time $O(m\gamma)$ [30]. Since the depth of an SCC-decomposition is at most the number of internal nodes plus one, and since Lemma 4 shows that the total number of nodes in a joint SCC-decomposition is $O(n \log n)$, it follows that the combined depth of all the SCC-decompositions that make up a joint SCC-decomposition is at most $O(n \log n)$, which proves the following lemma.

▶ **Lemma 5.** *The procedure* Build-Joint-SCC-Decomposition(G, S) *constructs a joint SCC-decomposition in time* $O(mn \log n)$.

To answer queries for the extended decremental SCC problem in constant time, we also construct and maintain an $n \times n$ matrix A such that $A[u, w]$ is the index of the SCC of $G \setminus \{w\}$ that contains u. Two vertices u and v are in the same SCC of $G \setminus \{w\}$ if and only if $A[u, w] = A[v, w]$. To avoid cluttering the pseudo-code we describe separately how A is maintained. In Build-Joint-SCC-Decomposition(G, S) we initialize A in the base case when we compute an SCC-decomposition T for a singleton $S = \{w\}$. Indeed, in this case w is the root of T, and the external nodes are exactly the SCCs of $G \setminus \{w\}$. Hence, for every vertex $u \in V(G) \setminus \{w\}$ we set $A[u, w]$ to the index of the SCC it is part of in $G \setminus \{w\}$.

Note that storing the matrix A takes space $O(n^2)$. The time spent initializing A is however dominated by the other work performed by the algorithm.

2.3 Deleting Edges from a Joint SCC-Decomposition

We next show how to maintain a joint SCC-decomposition under edge deletions. It is again instructive to consider the work performed by the naive algorithm that maintains n SCC-decompositions with distinct roots. If these are constructed as described in Section 2.2, then the SCC-decompositions will share many identical subtrees, and the work performed on these subtrees will be the same. In the joint SCC-decomposition such subtrees are shared, but otherwise the work performed is the same as the work performed for individual SCC-decompositions. We therefore use Łącki's algorithm [30] to delete edges from the individual SCC-decompositions, and we introduce a new procedure for handling the interface between the SCC-decompositions. We next briefly sketch Łącki's algorithm (see also [10, 30]).

Recall that each node ϕ of an SCC-decomposition T represents a strongly connected subgraph G_ϕ induced by the vertices in the subtree rooted at ϕ. If ϕ is an internal node of T, then the children of ϕ are the SCCs of $G_\phi \setminus \phi$. Łącki uses the following two operations to compactly represent edges among ϕ and its children.

▶ **Definition 6.** Let G be a graph. The *condensation* of G, denoted by Condense(G), is the graph obtained from G by contracting all its SCCs into single vertices. Let $v \in V(G)$. By Split(G, v) we denote the graph obtained from G by splitting v into two vertices: v_{in} and v_{out}. The in-edges of v are connected to v_{in} and the out-edges to v_{out}.

The two operations are often used together, and to simplify notation we use the shorthand $G_v^{\text{con}} = \textsc{Condense}(\textsc{Split}(G, v))$. The graph G_ϕ^{con} is stored with every internal node ϕ of the SCC-decomposition T. This introduces at most three copies of every vertex v of G: The two vertices v_{in} and v_{out} in G_v^{con}, and possibly a third vertex in the condensed graph of the parent of v in T. Moreover, every edge (u, v) appears in exactly one condensed graph, namely that of the lowest common ancestor of u and v in T, which we denote by $\text{LCA}(u, v)$. The combined space used for storing all the condensed graphs is thus $O(m + n)$.

To delete an edge (u, v), Łącki [30] locates $\phi = \text{LCA}(u, v)$, and deletes (u', v') from G_ϕ^{con}, where u' and v' are the vertices whose subtrees contain u and v. (He uses $O(m)$ space to store a pointer from every edge (u, v) to $\text{LCA}(u, v)$, enabling him to find the lowest common ancestor in constant time.) To preserve connectivity, he then checks whether u' and v' have non-zero out- and in-degrees, respectively, in G_ϕ^{con}. If this is not the case, then he repeatedly removes vertices with out- or in-degree zero and their adjacent edges from G_ϕ^{con}. All such vertices can be located, starting from u' and v', in time that is linear in the number of edges adjacent to the removed vertices. The corresponding children of ϕ are then moved up one level in T and are made siblings of ϕ. They are also inserted into $G_{par(\phi)}^{\text{con}}$, where $par(\phi)$ is the parent of ϕ, and their edges and the edges of ϕ in $G_{par(\phi)}^{\text{con}}$ are updated correspondingly. This can again be done in time linear in the number of edges in the original graph that are adjacent to vertices in the subtrees that are moved. The procedure is then repeated in $G_{par(\phi)}^{\text{con}}$. Since every vertex increases its level at most γ times, where γ is the initial depth of T, it follows that the total update time of the algorithm is at most $O(m\gamma)$.

We let $\textsc{Delete-Edge-from-SCC-decomposition}(T, u, v)$ be the procedure for deleting an edge (u, v) from an SCC-decomposition T. We also denote the recursive procedure for moving nodes $\phi_1, \ldots, \phi_k$ from being children of ϕ to being siblings of ϕ in T after an edge (u, v) is deleted by $\textsc{Fix-SCC-decomposition}(T, u, v, \phi, \{\phi_1, \ldots, \phi_k\})$. Both procedures return the resulting SCC-decomposition, or a collection of SCC-decompositions in case the graph is not strongly connected. (The pseudo-code appears in the full version of the paper.)

In a joint SCC-decomposition, vertices and edges may appear in multiple nodes as part of smaller SCC-decompositions. We therefore need to find every occurence of the edge that we wish to delete. We introduce a procedure $\textsc{Delete-Edge}(J, u, v)$ that does that by recursively searching through the nested joint SCC-decompositions and deleting (u, v) from the relevant SCC-decompositions. The procedure also handles the interface between SCC-decompositions. Note that deleting (u, v) from an SCC-decomposition T may cause the SCC corresponding to the root ϕ of T to break. The procedure $\textsc{Delete-Edge-from-SCC-decomposition}(T, u, v)$ will in this case return a collection of SCC-decompositions $\{T_1, \ldots, T_k\}$, one for each new SCC. Suppose $J = (J_1, J_2, S, \Phi)$. If T extends J_1 (resp. J_2), then it is an SCC-decomposition of the subgraph G_ϕ associated with some external node ϕ of J_1 (resp. J_2). ϕ is then itself a leaf of an SCC-decomposition T' in J_1 (resp. J_2). Moreover, when the SCC corresponding to ϕ breaks, then this leaf must be split into multiple leaves of T', one for each new SCC. Note however that the levels in T' of the involved vertices do not change after the split. We therefore cannot charge the work performed when splitting ϕ to the analysis by Łącki [30].

Let $\phi_1, \ldots, \phi_k$ be the roots of the SCC-decompositions $T_1, \ldots, T_k$ that are created when the deletion of (u, v) breaks the SCC G_ϕ. As mentioned above, we need to replace ϕ in T' by $\phi_1, \ldots, \phi_k$, which means that $\phi_1, \ldots, \phi_k$ should replace ϕ in $G_{par(\phi)}^{\text{con}}$, where $par(\phi)$ is the parent of ϕ in T'. To efficiently reconnect $\phi_1, \ldots, \phi_k$ in $G_{par(\phi)}^{\text{con}}$ we identify the vertex ϕ_i whose associated graph G_{ϕ_i} has the most vertices, and we then scan through all the vertices in the other graphs $G_{\phi_1}, \ldots, G_{\phi_{i-1}}, G_{\phi_{i+1}}, \ldots, G_{\phi_k}$ and reconnect their adjacent edges in $G_{par(\phi)}^{\text{con}}$ when relevant. The work performed is exactly the same as when Łącki

fixes an SCC-decomposition after an edge is removed. We can therefore call Fix-SCC-decomposition(T',u,v,ϕ,$\{\phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_k\}$). Note that this makes ϕ_i take over the role of ϕ. Also note that we provide the procedure with the end-points u and v of the edge that was deleted, since u and v are used as starting points for the search for disconnected vertices when propagating the update further up the tree.

Finally, observe that splitting the leaf ϕ of the SCC-decomposition T' may propagate all the way to the root of T' and break the SCC corresponding to T'. We therefore use a recursive procedure, Split-Leaf(J,u,v,ϕ,$\{\phi_1, \ldots, \phi_k\}$), to perform the split. Here u and v are again the end-points of the edge that was deleted.

▶ **Theorem 7.** *The total update time spent by* Delete-Edge(J, u, v) *in order to maintain a balanced joint SCC-decomposition under edge deletions is* $O(mn \log n)$.

Proof. We only sketch the proof, and refer to the full paper for additional details.

The time spent by Delete-Edge(J, u, v) consists of three parts: checking whether $\{u, v\} \subseteq V(G_\phi)$ for some $\phi \in \Phi$, the work performed while deleting edges from SCC-decompositions, and the work performed by Split-Leaf$(J_i, u, v, \phi, \{\phi_1, \ldots, \phi_k\})$, for $i \in \{1, 2\}$. To check in constant time whether $\{u, v\} \subseteq V(G_\phi)$, we maintain for each SCC-decomposition T an array on the vertices of the original graph G, such that $T\langle v\rangle =$ True if v appears in T, and $T\langle v\rangle =$ False otherwise. Storing these arrays takes up $O(n^2)$ space, and they are updated when the SCC of the root of an SCC-decomposition breaks.

Recall that Łącki [30] showed that the total initialization and update time of an SCC-decomposition is $O(m\gamma)$, where γ is the depth of the decomposition. By Lemma 4, the total number of nodes of the SCC-decompositions in J is $O(n \log n)$, and therefore the combined depth of the SCC-decompositions is also $O(n \log n)$. It follows that the time spent on Delete-Edge-from-SCC-decomposition(T_ϕ, u, v) is bounded by $O(mn \log n)$.

It remains to analyze the time spent on Split-Leaf$(J_i, u, v, \phi, \{\phi_1, \ldots, \phi_k\})$. Recall that Split-Leaf identifies the node ϕ_i that contains the most vertices from G, and then breaks off $\phi_1, \ldots, \phi_{i-1}, \phi_i, \ldots, \phi_k$ from ϕ. This means that ϕ is turned into ϕ_i, and that we do not scan through edges adjacent to vertices in ϕ_i. Since a split therefore moves vertices to new nodes of at most half the size, each vertex v can only be moved $O(\log n)$ times in one particular SCC-decomposition T by Split-Leaf. Each move takes time proportional to the number of edges adjacent to v, so the total time spent splitting leaves of T is at most $O(m \log n)$. Since, by Lemma 3, there are only $O(n)$ SCC-decompositions in J, it follows that the total time spent splitting leaves is $O(mn \log n)$. Furthermore, the time spent by Split-Leaf on fixing SCC-decompositions can be charged to the depth reduction of the vertices that are moved, and this part of the analysis is therefore the same as for Delete-Edge-from-SCC-decomposition(T_ϕ, u, v). ◀

Recall that we also maintain a matrix A for answering queries, where $A[u, w]$ is the index of the SCC of $G \setminus \{w\}$ that contains u. We again update A when we make changes to the topmost SCC-decompositions that each only contain a single internal node, i.e., when such a root ϕ gets a new child, or when G_ϕ breaks into multiple SCCs. The time spent updating A is dominated by the rest of the work that is performed by our algorithm, where we scan through all edges adjacent to vertices whose SCC is changed.

As described briefly in Section 2.3, Łącki's SCC-decomposition can be implemented such that is uses $O(m + n)$ space [30]. Since a balanced joint SCC-decomposition consists of $O(n)$ SCC-decompositions (Lemma 3), it follows that a naive implementation of our data structure uses $O(mn)$ space. In the full version of the paper we show how to obtain an alternative bound of $O(n^2 \log n)$. Doing so requires two observations:

- After an edge (u', v') is deleted from a condensed graph G_ϕ^{con}, the vertex u' has a path to ϕ_{in} if and only if u' has non-zero out-degree, and there is a path from ϕ_{out} to v' if and only if v' has non-zero in-degree. Instead of storing the edges of the condensed graphs we therefore store the in- and out-degrees of the vertices. To visit all neighbors of a vertex u' in G_ϕ^{con}, we then collect from the original graph G all edges adjacent to vertices in the subgraph of u', and we check for each edge whether the other end-point is part of G_ϕ^{con} and which vertex of G_ϕ^{con} it goes to. To do so we store pointers between the vertices of G and the vertices of the condensed graphs that they are part of. Since a joint SCC-decomposition contains $O(n \log n)$ nodes this takes $O(n^2 \log n)$ space.
- Since a joint SCC-decomposition contains $O(n \log n)$ nodes in total, we have time to visit all the nodes of an SCC-decomposition T when searching for the lowest common ancestor of two vertices u and v. This is done in a bottom-up fashion. We therefore do not need to store a pointer from every edge (u, v) to $\text{LCA}(u, v)$.

3 Applications

In this section we exploit the decremental joint SCC-decomposition to design decremental algorithms for various connectivity notions defined with respect to vertex or edge failures.

Maintaining Decrementally the Dominator Tree. We show how to maintain a dominator tree D of a flow graph G, rooted at a starting vertex s. We denote by $d(v)$ the parent of v in D. We first produce a flow graph G_s from G by adding an edge from each vertex $v \in V \setminus s$ to s. The addition of those edges has the following property. If a vertex u is not strongly connected to s in G_s then there is no path from s to u in G. Conversely, if a vertex u is not strongly connected to s in $G_s \setminus v$, while s and u are strongly connected in G_s, then all paths from s to u in G contain v. That is, v is a dominator of u in G.

We maintain decrementally a joint SCC-decomposition of G_s in a total of $O(mn \log n)$ time and $O(n^2 \log n)$ space. Therefore, for each vertex v we maintain the SCCs in $G \setminus v$. Let $v \neq s$: after the SCC containing s in $G \setminus v$ breaks, all the vertices that are not in the new SCC that contains s are dominated from v in G. We can report the newly dominated vertices from v in G in time proportional to their number. Therefore, after each edge deletion we need to process a batch $\mathcal{N}$ of incoming new dominance relations $N(v) = \{u_1, \ldots, u_k\}$, where $u_1, \ldots, u_k$ are dominated by v in G. We can process a batch of updates $\mathcal{N}$ in two phases as follows. For each vertex $u \neq s$ we keep a counter $\mathit{depth}(u)$ of the number of vertices that dominate u. For each dominance relation $N(v) \in \mathcal{N}$, we increase $\mathit{depth}(u)$ for each $u \in N(v)$. After this first phase ends, all vertices have updated counters. Then the new parent of each vertex u in D is the vertex with the largest counter among $d(u)$ (i.e., the parent of u in D before the edge insertion) and all vertices v such that $u \in N(v)$ and $N(v) \in \mathcal{N}$.

Now we bound the total time required to maintain the dominator tree. The running time of the above procedure, during the whole sequence of deletions, is bounded by the total size of all the sets $N(v)$. Note that any vertex can appear in a specific $N(v)$ set at most once during the deletion sequence. Hence, the total size of all the sets $N(v)$ is $O(n^2)$.

▶ **Lemma 8.** *The dominator tree of a directed graph G with start vertex s can be maintained decrementally in $O(mn \log n)$ total update time and $O(n^2 \log n)$ space, where m is the number of edges in the initial graph and n is the number of vertices.*

Maintaining Decrementally the Strong Bridges of the Graph. Let G be strongly connected: its strong bridges can be computed efficiently from a dominator tree D of G and a

dominator tree D^R of G^R, both rooted at the same start vertex s. We present a randomized algorithm that maintains such a pair of dominator trees in each SCC of a digraph in $O(mn \log n)$ total expected time, and $O(n^2 \log n)$ space. This allows us to maintain the set of strong bridges of a digraph in the same (expected) running time and space.

Maintaining Decrementally the 2-Edge-Connected Components. In this section we show how to maintain the 2-edge-connected components of directed graph. Two vertices w and z are 2-edge-connected if and only if there is no edge e such that w and z are not strongly connected in $G \setminus e$. A 2-edge-connected component is a maximal subset $B \subseteq V$, such that w and z are 2-edge-connected, for all $z, w \in B$. Therefore, a simple-minded algorithm for computing the 2-edge-connected components is the following. We start with the trivial partition $\mathcal{P}$ of the vertices that is equal to the set of SCCs of the graph. For every strong bridge e, we compute the SCCs $C_1, \ldots, C_k$ of $G \setminus e$ and we refine the maintained partition $\mathcal{P}$ according to the partition induced by the SCCs $C_1, \ldots, C_k$. After performing all refinements on $\mathcal{P}$ two vertices are in the same set if and only if we did not find an edge that separates them, which is exactly the definition of 2-edge-connected components.

Our algorithm is a dynamic version of the aforementioned simple-minded algorithm. That is, we maintain the SCCs of $G \setminus e$, for each strong bridge e, and refine the maintained partition $\mathcal{P}$ whenever we identify that $\mathcal{P}$ no longer contains the 2-vertex-connected components of G. We do this as follows. Assume that a component $C \in \mathcal{P}$ contains vertices in different SCCs of $G \setminus e$, for some e. Let $C_1, C_2, \ldots, C_k$ be the SCCs in $G \setminus e$. We replace C by $\{C \cap C_1\}, \ldots, \{C \cap C_k\}$. These refinements can be easily performed in $O(n)$ time, and therefore we spend total time $O(n^2)$ for all refinements throughout the algorithm.

In order to make our algorithm efficient we need to specify how to detect whether two 2-edge-connected vertices appear in different SCCs in $G \setminus e$, for some edge e. Whenever an SCC C in $G \setminus e$, breaks into k SCCs $C_1, \ldots, C_k$, for all SCCs C_i except the largest one we examine whether the components $C \in \mathcal{P}$ containing subsets of vertices of C_i are entirely contained in C_i. We develop machinery that allows us to list all vertices in the resulting SCCs $C_1, \ldots, C_k$ except the largest one, in time proportional to their number. The details can be found in the full paper. Each vertex can appear at most $\log n$ times in an SCC of $G \setminus e$, for some strong bridge e, that is not the largest after a big SCC breaks. This implies that we spend $O(n \log n)$ time for each graph $G \setminus e$ on testing whether an edge deletion leaves two (previously) 2-edge-connected vertices in different SCCs in $G \setminus e$, for some edge e. We show that at most $O(n)$ strong bridges can appear throughout any sequence of edge deletions. Thus we spend $O(n^2 \log n)$ time in total.

▶ **Lemma 9.** *The 2-edge-connected components of a digraph G can be maintained decrementally in $O(mn \log n)$ total expected time against an oblivious adversary, using $O(n^2 \log n)$ space, where m is the number of edges in the initial graph and n is the number of vertices.*

4 Conditional Lower Bound

In the following we give a conditional lower bound for the partially dynamic dominator tree problem. We show that there is no incremental nor decremental algorithm for maintaining the dominator tree that has total update time $O((mn)^{1-\epsilon})$ (for some constant $\epsilon > 0$) unless the OMv Conjecture [26] fails. This also holds for algorithms that do not explicitly maintain the tree, but are able to answer parent-queries. Formally, we prove the following statement.

▶ **Theorem 10.** *For any constant $\delta \in (0, 1/2]$ and any n and $m = \Theta(n^{1/(1-\delta)})$, there is no algorithm for maintaining a dominator tree under edge deletions/insertions allowing queries*

of the form "is x the parent of y in the dominator tree" that uses polynomial preprocessing time, total update time $u(m,n) = (mn)^{1-\epsilon}$ and query time $q(m) = m^{\delta-\epsilon}$ for some constant $\epsilon > 0$, unless the OMv conjecture fails.

Under this conditional lower bound, the running time of our algorithm is optimal up to sub-polynomial factors. We give the reduction for the decremental version of the problem. Hardness of the incremental version follows analogously.

In the online Boolean matrix-vector problem we are first given a Boolean $n \times n$ matrix M to preprocess. After the preprocessing, we are given a sequence of n-dimensional Boolean vectors $v^{(1)}, \ldots, v^{(n)}$ one by one. For each $1 \leq t \leq n$, we have to return the result of the matrix-vector multiplication $Mv^{(t)}$ before we are allowed to see the next vector $v^{(t+1)}$. The OMv Conjecture states that there is no algorithm that computes each matrix-vector product correctly (with high probability) and in total spends time $O(n^{3-\epsilon})$ for some constant $\epsilon > 0$.

We will not use the OMv problem directly as the starting point of our reduction. Instead we consider the following γ-OuMv problem (for a fixed $\gamma > 0$) and parameters n_1, n_2, and n_3 such that $n_1 = \lfloor n_2^\gamma \rfloor$: We are first given a Boolean $n_1 \times n_2$ matrix M to preprocess. After the preprocessing, we are given a sequence of pairs of n_1-dimensional Boolean vectors $(u^{(1)}, v^{(1)}), \ldots, (u^{(n_3)}, v^{(n_3)})$ one by one. For each $1 \leq t \leq n_3$, we have to return the result of the Boolean vector-matrix-vector multiplication $(u^{(t)})^\intercal M v^{(t)}$ before we see the next pair of vectors $(u^{(t+1)}, v^{(t+1)})$. It has been shown [26] that under the OMv Conjecture, there is no algorithm for this problem with polynomial preprocessing time and total processing time $O(n_1^{1-\epsilon_1} n_2^{1-\epsilon_2} n_3^{1-\epsilon_3})$ such that all ϵ_i are ≥ 0 and at least one ϵ_i is a constant > 0.

We now give the reduction from the γ-OuMv problem with $\gamma = \delta/(1-\delta)$ to the decremental dominator tree problem. In the following we denote by v_i the i-th entry of a vector i and by $M_{i,j}$ the entry at row i and column j of a matrix M.

Consider an instance of the γ-OuMv problem with parameters $n_1 = m^{1-\delta}$, $n_2 = m^\delta$, and $n_3 = m^{1-\delta}$. We preprocess the matrix M by constructing a graph $G^{(0)}$ with the set of vertices $V = \{s, x_1, \ldots, x_{n_3}, x_{n_3+1}, y_1, \ldots, y_{n_1}, z_1, \ldots, z_{n_2}\}$ and the following edges: (1) an edge (s, x_1), and, for every $1 \leq t \leq n_3$, an edge (x_t, x_{t+1}), (2) for every $1 \leq j \leq n_2$, an edge (t, z_j), (3) for every $1 \leq t \leq n_3$ and every $1 \leq i \leq n_1$, an edge (x_t, y_i), and (4) for every $1 \leq i \leq n_1$ and every $1 \leq j \leq n_2$, an edge (y_i, z_j) if and only if $M_{i,j} = 1$.

Whenever the algorithm is given the next pair of vectors $(u^{(t)}, v^{(t)})$, we first create a graph $G^{(t)}$ by performing the following edge deletions in $G^{(t-1)}$: If $t \geq 2$, we first delete all outgoing edges of x_{t-1}, except the one to x_t. Then (for any value of t), for every i such that $u_i^{(t)} = 0$ we delete the edge from x_t to y_i. Thus, for every $1 \leq i \leq n_1$, there will be an edge from x_t to y_i in $G^{(t)}$ if and only if $u_i^{(t)} = 1$. Having created $G^{(t)}$, we now, for every j such that $v_j^{(t)} = 1$, check whether x_t is the parent of z_j in the dominator tree. If this is the case for at least one j we return that $(u^{(t)})^\intercal M v^{(t)}$ is 1, otherwise we return 0.

The correctness of our reduction follows from the following lemma.

▶ **Lemma 11.** *For every $1 \leq t \leq n$, the j-th entry of $(u^{(t)})^\intercal M$ is 1 if and only if x_t is the immediate dominator of z_j in $G^{(t)}$*

Note that $(u^{(t)})^\intercal M v^{(t)}$ is 1 if and only if there is a j such that both the j-th entry of $u^{(t)}M$ as well as the j-th entry of $v^{(t)}$ are 1. Furthermore, x_t is the parent of z_j in the dominator tree if and only if x_t is an immediate dominator of z_j in the current graph. Therefore the lemma establishes the correctness of the reduction.

The initial graph $G^{(0)}$ has $n := \Theta(n_1 + n_2 + n_3) = \Theta(m^\delta + m^{1-\delta}) = \Theta(m^{1-\delta})$ vertices and $\Theta(n_1 n_2 + n_2 n_3) = \Theta(m)$ edges. The total number of parent-queries is $O(n_1 n_3) = m^{2(1-\delta)}$. Suppose the total update time of the decremental dominator tree algorithm is $O(u(m,n)) =$

$(mn)^{1-\epsilon}$ and its query time is $O(q(m)) = m^{\delta-\epsilon}$. Using the reduction above, we can thus solve the γ-OuMv problem for the parameters n_1, n_2, n_3 with polynomial preprocessing time and total update time $O(u(m,n) + m^{2(1-\delta)}q(m)) = O(u(m, m^{1-\delta}) + m^{2(1-\delta)}q(m)) = O(m^{2-\delta-\epsilon})$. Since $n_1 n_2 n_3 = m^{2-\delta}$, this means we would get an algorithm for the γ-OuMv problem with polynomial preprocessing time and total update time $O(n_1^{1-\epsilon_1} n_2^{1-\epsilon_2} n_3^{1-\epsilon_3})$ where at least one ϵ_i is a constant > 0. This contradicts the OMv Conjecture.

References

1. S. Allesina and A. Bodini. Who dominates whom in the ecosystem? Energy flow bottlenecks and cascading extinctions. *Journal of Theoretical Biology*, 230(3):351–358, 2004. doi:10.1016/j.jtbi.2004.05.009.
2. S. Alstrup, D. Harel, P. W. Lauridsen, and M. Thorup. Dominators in linear time. *SIAM Journal on Computing*, 28(6):2117–21132, 1999. doi:10.1137/S0097539797317263.
3. S. Alstrup and P. W. Lauridsen. A simple dynamic algorithm for maintaining a dominator tree. Technical Report 96-3, Department of Computer Science, University of Copenhagen, 1996.
4. M. E. Amyeen, W. K. Fuchs, I. Pomeranz, and V. Boppana. Fault equivalence identification using redundancy information and static and dynamic extraction. In *Proc. of the 19th IEEE VLSI Test Symposium*, pages 124–130, 2001. doi:10.1109/VTS.2001.923428.
5. S. Baswana, K. Choudhary, and L. Roditty. Fault tolerant reachability for directed graphs. In *Proc. of the 29th Int'l. Symposium on Distributed Computing (DISC)*, pages 528–543, 2015. doi:10.1007/978-3-662-48653-5_35.
6. S. Baswana, K. Choudhary, and L. Roditty. Fault tolerant reachability subgraph: Generic and optimal. In *Proc. of the 48th ACM Symposium on Theory of Computing (STOC)*, pages 509–518, 2016. doi:10.1145/2897518.2897648.
7. A. L. Buchsbaum, L. Georgiadis, H. Kaplan, A. Rogers, R. E. Tarjan, and J. R. Westbrook. Linear-time algorithms for dominators and other path-evaluation problems. *SIAM Journal on Computing*, 38(4):1533–1573, 2008. doi:10.1137/070693217.
8. A. L. Buchsbaum, H. Kaplan, A. Rogers, and J. R. Westbrook. A new, simpler linear-time dominators algorithm. *ACM Transactions on Programming Languages and Systems*, 20(6):1265–1296, 1998. doi:10.1145/295656.295663.
9. M. D. Carroll and B. G. Ryder. Incremental data flow analysis via dominator and attribute update. In *Proc. of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 274–284, 1988. doi:10.1145/73560.73584.
10. S. Chechik, T. D. Hansen, G. F. Italiano, J. Lacki, and N. Parotsidis. Decremental single-source reachability and strongly connected components in $\tilde{O}(m\sqrt{n})$ total update time. In *Proc. of the 57th IEEE Symposium on Foundations of Computer Science(FOCS)*, pages 315–324, 2016. doi:10.1109/FOCS.2016.42.
11. S. Cicerone, D. Frigioni, U. Nanni, and F. Pugliese. A uniform approach to semi-dynamic problems on digraphs. *Theorical Computer Science*, 203:69–90, August 1998. doi:10.1016/S0304-3975(97)00288-0.
12. R. Cytron, J. Ferrante, B. K. Rosen, M. N. Wegman, and F. K. Zadeck. Efficiently computing static single assignment form and the control dependence graph. *ACM Transactions on Programming Languages and Systems*, 13(4):451–490, 1991. doi:10.1145/115372.115320.
13. W. Di Luigi, L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-connectivity in directed graphs: An experimental study. In *Proc. of the 17th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 173–187, 2015. doi:10.1137/1.9781611973754.15.

14 D. Eppstein, Z. Galil, and G. F. Italiano. Dynamic graph algorithms. In M. J. Atallah and M. Blanton, editors, *Algorithms and Theory of Computation Handbook, 2nd Edition, Vol. 1*, pages 9.1–9.28. CRC Press, 2009.

15 K. Gargi. A sparse algorithm for predicated global value numbering. *SIGPLAN Not.*, 37(5):45–56, 2002. doi:10.1145/543552.512536.

16 L. Georgiadis. Testing 2-vertex connectivity and computing pairs of vertex-disjoint *s*-*t* paths in digraphs. In *Proc. of the 37th Int'l. Coll. on Automata, Languages, and Programming (ICALP)*, pages 738–749, 2010. doi:10.1007/978-3-642-14165-2_62.

17 L. Georgiadis. Approximating the smallest 2-vertex connected spanning subgraph of a directed graph. In *Proc. of the 19th European Symposium on Algorithms (ESA)*, pages 13–24, 2011. doi:10.1007/978-3-642-23719-5_2.

18 L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-vertex connectivity in directed graphs. In *Proc. of the 42nd Int'l. Coll. on Automata, Languages, and Programming (ICALP)*, pages 605–616, 2015. doi:10.1007/978-3-662-47672-7_49.

19 L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-edge connectivity in directed graphs. *ACM Transactions on Algorithms*, 13(1):9:1–9:24, 2016. doi:10.1145/2968448.

20 L. Georgiadis, G. F. Italiano, L. Laura, and F. Santaroni. An experimental study of dynamic dominators. In *Proc. of the 20th European Symposium on Algorithms (ESA)*, pages 491–502, 2012. doi:10.1007/978-3-642-33090-2_43.

21 L. Georgiadis, G. F. Italiano, and N. Parotsidis. Incremental strong connectivity and 2-connectivity in directed graphs. Manuscript, 2017.

22 L. Georgiadis, G. F. Italiano, and N. Parotsidis. Strong connectivity in directed graphs under failures, with applications. In *Proc. of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1880–1899, 2017. doi:10.1137/1.9781611974782.123.

23 L. Georgiadis and R. E. Tarjan. Finding dominators revisited. In *Proc. of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 869–878, 2004.

24 M. Gomez-Rodriguez, L. Song, N. Du, H. Zha, and B. Schölkopf. Influence estimation and maximization in continuous-time diffusion networks. *ACM Transactions on Information Systems*, 34(2):9:1–9:33, 2016. doi:10.1145/2824253.

25 M. Henzinger, S. Krinninger, and V. Loitzenbauer. Finding 2-edge and 2-vertex strongly connected components in quadratic time. In *Proc. of the 42nd Int'l. Coll. on Automata, Languages, and Programming (ICALP)*, pages 713–724, 2015. doi:10.1007/978-3-662-47672-7_58.

26 M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proc. of the 47th ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2015. doi:10.1145/2746539.2746609.

27 G. F. Italiano, L. Laura, and F. Santaroni. Finding strong bridges and strong articulation points in linear time. *Theoretical Computer Science*, 447:74–84, 2012. doi:10.1016/j.tcs.2011.11.011.

28 R. Jaberi. Computing the 2-blocks of directed graphs. *RAIRO – Theoretical Informatics and Applications*, 49(2):93–119, 2015. doi:10.1051/ita/2015001.

29 R. Jaberi. On computing the 2-vertex-connected components of directed graphs. *Discrete Applied Mathematics*, 204:164–172, 2016. doi:10.1016/j.dam.2015.10.001.

30 J. Lacki. Improved deterministic algorithms for decremental reachability and strongly connected components. *ACM Transactions on Algorithms*, 9(3):27, 2013. doi:10.1145/2483699.2483707.

31 E. K. Maxwell, G. Back, and N. Ramakrishnan. Diagnosing memory leaks using graph mining on heap dumps. In *Proc. of the 16th ACM SIGKDD Int'l. Con. on Knowledge Discovery and Data Mining (KDD)*, pages 115–124, 2010. doi:10.1145/1835804.1835822.

32 M. Mihalák, P. Uznański, and P. Yordanov. Prime factorization of the Kirchhoff polynomial: Compact enumeration of arborescences. In *Proc. of the SIAM Analytic Algorithmics and Combinatorics (ANALCO)*, pages 93–105, 2016. doi:10.1137/1.9781611974324.10.

33 K. Patakakis, L. Georgiadis, and V. A. Tatsis. Dynamic dominators in practice. In *Proc. of the 16th Panhellenic Conference on Informatics (PCI)*, pages 100–104, 2011. doi:10.1109/PCI.2011.28.

34 L. Quesada, P. Van Roy, Y. Deville, and R. Collet. Using dominators for solving constrained path problems. In *Proc. of the 8th International Conference on Practical Aspects of Declarative Languages (PADL)*, pages 73–87, 2006. doi:10.1007/11603023_6.

35 G. Ramalingam and T. Reps. An incremental algorithm for maintaining the dominator tree of a reducible flowgraph. In *Proc. of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 287–296, 1994. doi:10.1145/174675.177905.

36 V. C. Sreedhar, G. R. Gao, and Y. Lee. Incremental computation of dominator trees. *ACM Transactions on Programming Languages and Systems*, 19:239–252, 1997. doi:10.1145/202529.202531.

37 R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

General Bounds for Incremental Maximization*

Aaron Bernstein[1], Yann Disser[2], and Martin Groß[3]

1 **TU Berlin, Berlin, Germany**
bernstei@gmail.com
2 **TU Darmstadt, Darmstadt, Germany**†
disser@mathematik.tu-darmstadt.de
3 **University of Waterloo, Waterloo, Canada**‡
mgrob@uwaterloo.ca

Abstract

We propose a theoretical framework to capture incremental solutions to cardinality constrained maximization problems. The defining characteristic of our framework is that the cardinality/support of the solution is bounded by a value $k \in \mathbb{N}$ that grows over time, and we allow the solution to be extended one element at a time. We investigate the best-possible competitive ratio of such an incremental solution, i.e., the worst ratio over all k between the incremental solution after k steps and an optimum solution of cardinality k. We define a large class of problems that contains many important cardinality constrained maximization problems like maximum matching, knapsack, and packing/covering problems. We provide a general 2.618-competitive incremental algorithm for this class of problems, and show that no algorithm can have competitive ratio below 2.18 in general.

In the second part of the paper, we focus on the inherently incremental greedy algorithm that increases the objective value as much as possible in each step. This algorithm is known to be 1.58-competitive for submodular objective functions, but it has unbounded competitive ratio for the class of incremental problems mentioned above. We define a relaxed submodularity condition for the objective function, capturing problems like maximum (weighted) (b-)matching and a variant of the maximum flow problem. We show that the greedy algorithm has competitive ratio (exactly) 2.313 for the class of problems that satisfy this relaxed submodularity condition.

Note that our upper bounds on the competitive ratios translate to approximation ratios for the underlying cardinality constrained problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases incremental optimization, maximization problems, greedy algorithm, competitive analysis, cardinality constraint

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.43

1 Introduction

Practical solutions to optimization problems are often inherently *incremental* in the sense that they evolve historically instead of being established in a one-shot fashion. This is especially true when solutions are expensive and need time and repeated investments to be implemented, for example when optimizing the layout of logistics and other infrastructures.

* A full version of the paper is available at https://arxiv.org/abs/1705.10253.
† Supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School CE at TU Darmstadt.
‡ Supported by the German Research Foundation (DFG) within project A07 of CRC TRR 154.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 43; pp. 43:1–43:14


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

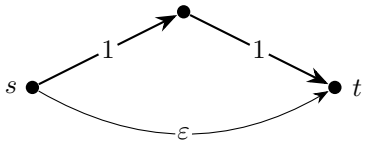


Figure 1 Example showing that the s-t-flow problem does not always admit good incremental solutions, where $\varepsilon > 0$ is arbitrarily small.

In this paper, we propose a theoretical framework to capture *incremental maximization* problems in some generality.

We describe an incremental problem by a set U containing the possible elements of a solution, and an objective function $f\colon 2^U \to \mathbb{R}^+$ that assigns to each solution $S \subseteq U$ some non-negative value $f(S)$. We consider problems of the form

$$\begin{aligned} &\max\ f(S) \\ &\text{s.t. } |S| \leq k \\ &\quad S \subseteq U, \end{aligned} \tag{1}$$

where $k \in \mathbb{N}$ grows over time.

An incremental solution $\vec{S}$ is given by an order $\{s_1, s_2, \dots\} := U$ in which the elements of U are to be added to the solution over time. A good incremental solution needs to provide a good solution after k steps, for every k, compared to an optimum solution $S_k^\star$ with k elements, where we let $S_k^\star \in \arg\max_{S\subseteq U, |S|=k} f(S)$ and $f_k^\star := f(S_k^\star)$. Formally, we measure the quality of an incremental solution by its *competitive ratio*. For $\vec{S}_k := \{s_1, \dots, s_k\} \subseteq U$ being the first k elements of $\vec{S}$, we say that $\vec{S}$ is (strictly) ρ-competitive if

$$\max_{k\in\{1,\dots,|U|\}} \frac{f_k^\star}{f(\vec{S}_k)} \leq \rho.$$

An algorithm is called ρ-competitive if it always produces a ρ-competitive solution, and its *competitive ratio* is the infimum over all $\rho \geq 1$ such that it is ρ-competitive. Notice that we do not require the algorithm to run in polynomial time.

While all cardinality constrained optimization problems can be viewed in an incremental setting, clearly not all such problems admit good incremental solutions. For example, consider a cardinality constrained formulation of the classical maximum s-t-flow problem: For a given graph $G = (V, E)$, two vertices $s, t \in V$ and capacities $u\colon E \to \mathbb{R}^+$, we ask for a subset $E' \subseteq E$ of cardinality $k \in \mathbb{N}$ such that the maximum flow in the subgraph (V, E') is maximized. The example in Figure 1 shows that we cannot hope for an incremental solution that is simultaneously close to optimal for cardinalities 1 and 2.

In order to derive general bounds on the competitive ratio of incremental problems, we need to restrict the class of objective functions f that we consider. Intuitively, the unbounded competitive ratio in the flow example comes from the fact that we have to invest in the s-t-path of capacity 1 as soon as possible, but this path only yields its payoff once it is completed after two steps.

In order to prevent this and similar behaviors, we require f to be monotone (i.e., $f(S) \leq f(T)$ if $S \subseteq T$) and sub-additive (i.e., $f(S) + f(T) \geq f(S \cup T)$). Many important optimization problems satisfy these weak conditions, and we give a short list of examples below. We will see that all these (and many more) problems admit incremental solutions with a bounded competitive ratio. More specifically, we develop a general 2.618-competitive

incremental algorithm that can be applied to a broad class of problems, including all problems mentioned below. We illustrate in detail how to apply our model to obtain an incremental variant of the matching problem, and then list incremental versions of other important problems that are obtained analogously.

- Maximum Weighted Matching: Consider a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_{\geq 0}$. If we think of edges as *potential* connections and edge weights as *potential* payoffs, then it is not enough to find the final matching because we cannot construct the edges all at once: the goal is to find a sequence of edges that achieves a high pay-off in the short, the medium, and the long term. In terms of our formal framework, we add edges to a set S one at a time with $U = E$ and $f(S)$ is the maximum weight of a matching $M \subseteq S$. In order to be ρ-competitive, we need that, after k steps for every k, our solution S of cardinality k is no worse than a factor of ρ away from the optimum solution of cardinality k, i.e., $f(S) \geq f(S^\star_k)/\rho$.
  This model captures the setting where the infrastructure (e.g. the matching, the knapsack, the covering, or the flow) must be built up over time. The online model would be too restrictive in this setting because here we know our options in advance. Note that, as we add more edges, the set of edges S only needs to contain a large matching M, but does not have to be a matching itself; The matching M can change to an arbitrary subset of S from one cardinality to the next and does not have to stay consistent. This ensures that $f(S)$ is monotonically increasing, and is in keeping with the infrastructures setting where the potential regret present in the online model does not apply: building more infrastructure can only help, since once it is built, we can change how it is used. Accordingly, in all the problems below the set S does not have to be a valid solution to the cardinality constrained problem at hand, but rather needs to *contain* a good solution as a subset. The objective $f(S)$ is consistently defined to be the value of the best solution that is a subset of S. Notice that this approach can easily be generalized to Maximum b-Matching.
- Set Packing: Given a set of weighted sets $\mathcal{X}$ we ask for an incremental subset $S \subseteq \mathcal{X}$ where $f(S)$ is the maximum weight of mutually disjoint subsets in S. This problem captures many well-known problems such as Maximum Hypergraph Matching and Maximum Independent Set.
- Maximum Coverage: Given a set of weighted sets $\mathcal{X} \subseteq 2^U$ over an universe of elements U, we ask for an incremental subset $S \subseteq \mathcal{X}$, where $f(S)$ is the weight of elements in $\bigcup_{X \in S} X$. This problem captures maximization versions of clustering and location problems. We can include opening costs $c \colon \mathcal{X} \to \mathbb{R}_{\geq 0}$ by letting $f(S)$ be the maximum over all subsets $S' \subseteq S$ of the number (or weight) of the sets in S' minus their opening costs.
- Knapsack: Given a set X of items, associated sizes $s \colon X \to \mathbb{R}_{\geq 0}$ and values $v \colon X \to \mathbb{R}_{\geq 0}$, and a knapsack of capacity 1, we ask for an incremental subset $S \subseteq X$, where $f(S)$ is the largest value $\sum_{x \in S'} v(x)$ of any subset $S' \subseteq S$ with $\sum_{x \in S'} s(x) \leq 1$. This problem can be generalized to Multi-Dimensional Knapsack by letting item sizes be vectors and letting the knapsack have a capacity in every dimension.
- Disjoint Paths: Given a graph $G = (V, E)$, a set of pairs $\mathcal{X} \subseteq V^2$ with weights $w \colon \mathcal{X} \to \mathbb{R}_{\geq 0}$, we ask for an incremental subset $S \subseteq \mathcal{X}$, where $f(S)$ is the maximum weight of a subset $S' \subseteq S$, such that G contains mutually disjoint paths between every pair in S'.
- Maximum Bridge-Flow: We argued above that the maximum s-t-flow problem is not amenable to the incremental setting because it does not pay off to build paths partially.

To overcome this, we consider a natural restriction of the flow problem where most edges are freely available to be used, and only the edges of a directed s-t-cut need to be built incrementally. If the directed cut has no backward edges, every s-t-path contains exactly one edge that needs to be built, and we never have to invest multiple steps to establish a single path. This problem captures logistical problems where links need to be established between two clusters, like when bridges need to be built across a river, cables across an ocean, or when warehouses need to be opened in a supplier-warehouse-consumer network. Formally, given a directed graph $G = (V, E)$ with capacities $u\colon E \to \mathbb{R}$, vertices $s, t \in V$, and a directed s-t-cut $C \subseteq E$ induced by the partition (U, W) of V such that the directed cut induced by (W, U) is empty, we ask for an incremental subset $S \subseteq C$ where $f(S)$ is the value of a maximum flow in the subgraph $(V, E \setminus (C \setminus S))$.

It is easy to verify that all the problems mentioned above (and many more) indeed have a monotone and sub-additive objective function. In addition, each one of these problems satisfies the following property: For every $S \subseteq U$, there exists $s \in S$ with $f(S \setminus \{s\}) \geq f(S) - f(S)/|S|$. We call this property the *accountability* property – to our knowledge, it has not been named before. Intuitively, this property ensures that the value of a set $S \subseteq U$ is the sum of individual contributions of its elements, and there cannot be additional value that emerges only when certain elements of U combine. While it is easy to formulate artificial problems that have monotonicity and sub-additivity but no accountability, we were not able to identify any natural problems of this kind. This justifies to add accountability to the list of properties that we require of incremental problems.

▶ **Definition 1.** Given a set of elements U, and a function $f\colon 2^U \to \mathbb{R}$, we say that the function f is *incremental* if it satisfies the following properties for every $S, T \subseteq U$:

1. (monotonicity): $S \subseteq T \Rightarrow f(S) \leq f(T)$,
2. (sub-additivity): $f(S) + f(T) \geq f(S \cup T)$,
3. (accountability): $\exists s \in S\colon f(S \setminus \{s\}) \geq f(S) - f(S)/|S|$.

We say that a cardinality constrained problem with increasing cardinality (eq. (1)) is *incremental* if its objective function is incremental.

Observe that a ρ-competitive incremental algorithm immediately yields a ρ-approximation algorithm for the underlying cardinality constrained problem, with the caveat that the resulting approximation algorithm might not be efficient since we make no demands on the runtime of the incremental algorithm. The converse is rarely the case since approximation algorithms usually do not construct their solution in incremental fashion. A prominent exception are *greedy* algorithms that are inherently incremental in the sense that they pick elements one-by-one such that each pick increases the objective by the maximum amount possible. This type of a greedy algorithm has been studied as an approximation algorithm for many cardinality constrained problems, and approximation ratios translate immediately to competitive ratios for the incremental version of the corresponding problem. In particular, the greedy algorithm is known to have competitive ratio (exactly) $\frac{e}{e-1} \approx 1.58$ if the objective function f is monotone and submodular [27]. Note, however, that of all the incremental problems listed above, only Maximum Coverage (without opening costs) has a submodular objective function. It is also known that if we relax the submodularity requirement and allow f to be the minimum of two monotone (sub-)modular functions, the greedy algorithm can be arbitrarily bad [21]. We provide a different relaxation of submodularity that captures Maximum (Weighted) (b-)Matching and Maximum Bridge-Flow, and where the greedy algorithm has a bounded competitive/approximation ratio.

Our Results. As our first result, we show that every incremental problem admits a bounded competitive ratio.

▶ **Theorem 2.** *Every incremental problem admits a $(1+\varphi)$-competitive algorithm, where φ is the golden ratio and $(1+\varphi) \approx 2.618$. No general deterministic algorithm for this class of problems has a competitive ratio of 2.18 or better.*

Again, note that we make no guarantees regarding the running time of our incremental algorithm. In fact, our algorithm relies on the ability to compute the optimum of the underlying cardinality constrained problem for increasing cardinalities. If we can provide an efficient approximation of this optimum, we get an efficient incremental algorithm in the following sense.

▶ **Corollary 3.** *If there is a polynomial time α-approximation algorithm for a cardinality constrained problem with incremental objective function, then we can design a polynomial time $\alpha(1+\varphi)$-competitive incremental algorithm.*

We also analyze the approximation/competitive ratio of the greedy algorithm. We observe that for many incremental problems like Knapsack, Maximum Independent Set, and Disjoint Paths, the greedy algorithm has an unbounded competitive ratio. On the other hand, we define a relaxation of submodularity called *α-augmentable* under which the greedy algorithm has a bounded competitive ratio. In particular, this relaxation captures our cardinality constrained versions of Maximum (Weighted) (b-)Matching and Maximum Bridge-Flow, where the incremental set S need not be feasible but only contain a good feasible subset. We get the following result, where the tight lower bound for $\alpha = 2$ is obtained for Maximum Bridge-Flow. Notice that for $\alpha = 1$, we obtain the $\frac{e}{e-1} \approx 1.58$ bound that is known for submodular functions. For $\alpha = 2$, the bound is $\frac{e^2}{e^2-1} \approx 2.313$.

▶ **Theorem 4.** *For every cardinality constrained problem with an α-augmentable objective (defined below), the greedy algorithm has approximation/competitive ratio $\alpha\frac{e^\alpha}{e^\alpha-1}$. This bound is tight for the greedy algorithm on problems with 2-augmentable objectives, which includes* Maximum (Weighted) (b-)Matching *and* Maximum Bridge-Flow.

We emphasize that the families of instances we construct to obtain the lower bounds in Theorems 2 and 4 require the number of elements to tend to infinity, since it takes time for incremental solutions to sufficiently fall behind the optimum solution.

Related Work. Most work on incremental settings has focused on cardinality constrained *minimization* problems. A prominent exception is the robust matching problem, introduced by Hassin and Rubinstein [16]. This problem asks for a weighted matching M with the property that, for every value k, the total weight of the $\min(k, |M|)$ heaviest edges of M comes close to the weight of a maximum weight matching of cardinality k. Note that this differs from our definition of incremental matchings in that the robust matching problem demands that the "incremental" solution consists of a matching, while we allow any edge set that contains a heavy matching as a subset. Since their model is more strict, all of the following competitive ratios carry over to our setting. Note that, in contrast to our setting, the objective function of the robust matching problem is submodular, and hence the greedy algorithm has competitive ratio at most $\frac{e}{e-1} \approx 1.58$ [27]. Hassin and Rubinstein [16] gave an improved, deterministic algorithm that achieves competitive ratio $\sqrt{2} \approx 1.414$. They also give a tight example for the $\sqrt{2}$ ratio, which also works in our incremental setting. Fujita et al. [11] extended this result to matroid intersection, and Kakimura and Makino [18] showed

that every independence system allows for a $\sqrt{\mu}$-competitive solution, with μ being the extendibility of the system. Matuschke at al. [24] describe a randomized algorithm for this problem that, under the assumption that the adversary does not know the outcome of the randomness, has competitive ratio $\ln(4) \approx 1.386$.

A variant of the knapsack problem with a similar notion of robustness was proposed by Kakimura et al. [19]. In this problem a knapsack solution needs to be computed, such that, for every k, the value of the k most valuable items in the knapsack compares well with the optimum solution using k items, for every k. Kakimura et al. [19] restrict themselves to polynomial time algorithms and show that under this restriction a bounded competitive ratio is possible only if the rank quotient of the knapsack system is bounded. In contrast, our results show that if we do not restrict the running time and if we only require our solution to *contain* a good packing with k items for every k, then we can be $(1+\varphi)$-competitive using our generic algorithm, even for generalizations like Multi-Dimensional Knapsack. If we restrict the running time and use the well-known PTAS for the knapsack problem [17, 22], we still get a $(1+\varphi)(1+\varepsilon)$-competitive algorithm. Megow and Mestre [25] and Disser et al. [7] considered another variant of the knapsack problem that asks for an order in which to pack the items that works well for every knapsack capacity. Kobayashi and Takizawa [20] study randomized strategies for cardinality robustness in the knapsack problem.

Hartline and Sharp [15] considered an incremental variant of the maximum flow problem where capacities increase over time. This is in contrast to our framework where the cardinality of the solution increases.

Incremental solutions for cardinality constrained minimization problems have been studied extensively, in particular for clustering [3, 6], k-median [4, 10, 26], minimium spanning tree [1, 12], and facility location [13]. An important result in this domain is the incremental framework given by Lin et al. [23]. This general framework allows to devise algorithms for every incremental minimization problem for which a suitable augmentation subroutine can be formulated. Lin et al. [23] used their framework to match or improve many of the known specialized bounds for the problems above and to derive new bounds for covering problems. In contrast to their result, our incremental framework allows for a general algorithm that works out-of-the-box for a broad class of incremental maximization problems and yields a constant (relatively small) competitive ratio.

Abstractly, incremental problems can be seen as optimization problems under uncertainty. Various approaches to handling uncertain input data have been proposed, ranging from robust and stochastic optimization to streaming and exploration. On this level, incremental problems can be seen as a special case of online optimization problems, i.e., problems where the input data arrives over time (see [2, 9]). Whereas online optimization in general assumes adversarial input, incremental problems restrict the freedom of the adversary to deciding when to stop, i.e., the adversary may choose the cardinality k while all other data is fixed and known to the algorithm. Online problems with such a "non-adaptive" adversary have been studied in other contexts [5, 8, 14]. Note that online problems demand irrevocable decisions in every time step – a requirement that may be overly restrictive in many settings where solutions develop over a long time period. In contrast, our incremental model only requires a growing solution "infrastructure" and allows the actual solution to change arbitrarily over time within this infrastructure.

2 A competitive algorithm for incremental problems

In this section, we show the second part of Theorem 2, i.e., we give an incremental algorithm that is $(1+\varphi \approx 2.618)$-competitive for all incremental problems. For convenience, we define

the *density* δ_S of a set $S \subseteq U$ via $\delta_S := f(S)/|S|$, and we let $\delta_k^\star := \delta_{S_k^\star}$ denote the optimum density for cardinality k. Our algorithm relies on the following two observations that follow from the accountability of the objective function.

▶ **Lemma 5.** *In every incremental problem and for every cardinality k, there is an ordering $\vec{S}_k^\star := \{s_1^\star, s_2^\star, \dots, s_k^\star\} := S_k^\star$, such that $\delta_{\{s_1^\star,\dots,s_i^\star\}} \geq \delta_{\{s_1^\star,\dots,s_{i+1}^\star\}}$ for all $i \in \{1,\dots,k-1\}$. We say that $\vec{S}_k^\star$ is a* greedy order *of $S_k^\star$.*

Proof. By accountability of f, there is an element $s_k^\star \in S_k^\star$ for which

$$\delta_{S_k^\star \setminus \{s_k^\star\}} = \frac{f(S_k^\star \setminus \{s_k^\star\})}{k-1} \geq \frac{f(S_k^\star)}{k} = \delta_{S_k^\star}.$$

We can repeat this argument for $s_{k-1}^\star \in S_k^\star \setminus \{s_k^\star\}$, $s_{k-2}^\star \in S_k^\star \setminus \{s_k^\star, s_{k-1}^\star\}$, etc. to obtain the desired ordering $\vec{S}_k^\star$. ◀

▶ **Lemma 6.** *In every incremental problem and for every $1 \leq k' \leq k$ we have $\delta_{k'}^\star \geq \delta_k^\star$.*

Proof. Fix any cardinality $k > 1$. By accountability of the objective function f, there is an element $s^\star \in S_k^\star$ with

$$\delta_k^\star = \frac{f(S_k^\star)}{k} \leq \frac{f(S_k^\star \setminus \{s^\star\})}{k-1} \leq \frac{S_{k-1}^\star}{k-1} = \delta_{k-1}^\star.$$

It follows that $\delta_k^\star$ is monotonically decreasing in k. ◀

Now, we define $k_0 := 1$ and $k_i := \lceil (1+\varphi) k_{i-1} \rceil$ for all positive integers i. Our algorithm operates in phases $i \in \{0, 1, \dots\}$. In each phase i, we add the elements of the optimum solution $S_{k_i}^\star$ of cardinality k_i to our incremental solution in greedy order (Lemma 5). Note that we allow the algorithm to add elements multiple times (without effect) in order to not complicate the analysis needlessly (of course we would only improve the algorithm by skipping over duplicates). In the following, we denote by t_i the number of steps (possibly without effect) until the end of phase i, i.e., we let $t_0 := k_0$ and $t_i := t_{i-1} + k_i$.

▶ **Lemma 7.** *For every phase $i \in \{0, 1, \dots\}$, we have $t_i \leq \varphi k_i$.*

Proof. We use induction over i, with the case $i = 0$ being trivial, since $t_0 = k_0$. Now assume that $t_{i-1} \leq \varphi k_{i-1}$ for some $i \geq 1$. Using the property $\frac{\varphi}{\varphi+1} = \varphi - 1$ of the golden ratio, we get

$$t_i = t_{i-1} + k_i \leq \varphi k_{i-1} + k_i \leq \frac{\varphi}{\varphi+1} k_i + k_i = \varphi k_i.$$ ◀

Finally, we show the solution $\vec{S}$ computed by our algorithm is $(1+\varphi)$-competitive.

▶ **Theorem 8.** *For every cardinality k, we have $f(\vec{S}_k) \geq f_k^\star/(1+\varphi)$.*

Proof. We use induction over k. The claim is true for $k = t_0 = 1$, since $\vec{S}_1 = S_1^\star$ by definition of the algorithm. For the inductive step, we prove that if the claim is true for $k = t_{i-1}$, then it remains true for all $k \in \{t_{i-1}+1, \dots, t_i\}$. Recall that $k_i = \lceil (1+\varphi) k_{i-1} \rceil$. By Lemma 7, we have

$$t_{i-1} \leq \varphi k_{i-1} < k_i < t_{i-1} + k_i = t_i,$$

and we can therefore distinguish the following cases.

Case 1: $t_{i-1} < k < k_i$. Since $k > t_{i-1}$, our algorithm has already completed phase $i-1$ and added all elements of $S^\star_{k_{i-1}}$, so we have $f(\vec{S}_k) \geq f^\star_{k_{i-1}}$. Because k is an integer and $k < k_i = \lceil (1+\varphi)k_{i-1} \rceil$, we have that $k < (1+\varphi)k_{i-1}$. By Lemma 6, we thus have

$$f^\star_k = \delta^\star_k \cdot k < \delta^\star_{k_{i-1}} \cdot (1+\varphi)k_{i-1} = (1+\varphi) f^\star_{k_{i-1}} \leq (1+\varphi) f(\vec{S}_k).$$

Case 2: $k_i \leq k \leq t_i$. At time k, our algorithm has already completed the first $k - t_{i-1}$ elements of $S^\star_k$. Since the algorithm adds the elements of $S^\star_{k_i}$ in greedy order, we have $f(\vec{S}_k) \geq (k - t_{i-1})\delta^\star_{k_i}$. On the other hand, since $k \geq k_i$, by Lemma 6 we have $f^\star_k = k \cdot \delta^\star_k \leq k \cdot \delta^\star_{k_i}$. In order to complete the proof, it is thus sufficient to show that $k \leq (1+\varphi)(k - t_{i-1})$. To see this, let $k = k_i + k'$ for some non-negative integer k'. Because t_{i-1} is integral, Lemma 7 implies $t_{i-1} \leq \lfloor \varphi k_{i-1} \rfloor$. Since φ is irrational and k_{i-1} is integral, φk_{i-1} cannot be integral, thus

$$k - t_{i-1} = k' + k_i - t_{i-1} \geq k' + \lceil (1+\varphi)k_{i-1} \rceil - \lfloor \varphi k_{i-1} \rfloor = k' + k_{i-1} + 1.$$

This completes the proof, since

$$(1+\varphi)(k - t_{i-1}) \geq (1+\varphi)(k' + k_{i-1} + 1) > k' + (1+\varphi)k_{i-1} + 1 \geq k' + k_i = k. \qquad \blacktriangleleft$$

Corollary 3 follows if we replace $S^\star_{k_i}$ by an α-approximate solution for cardinality k_i.

3 Lower bound on the best-possible competitive ratio

In this section, we show the second part of Theorem 2, i.e., we give a lower bound on the best-possible competitive ratio for the maximization of incremental problems. For this purpose, we define the Region Choosing problem. In this problem, we are given N disjoint sets $R_1, \ldots, R_N$, called *regions*, with region R_i containing i elements with a value of $\delta(i)$ each. We say that $\delta(i)$ is the *density* of region R_i. The total value of all elements in the region R_i is $v(i) := i \cdot \delta(i)$ for all $i \in \{1, \ldots, N\}$.

The objective is to compute an incremental solution $S \subseteq U := \bigcup_{i=1}^N R_i$ such that the maximum value of the items from a single region in S is large. Formally, the objective function is given by $f(S) := \max_{i \in \{1,\ldots,N\}} |R_i \cap S| \cdot v(i)$.

▶ **Observation 9.** Region Choosing *is an incremental problem.*[1]

For our lower bound, we set $\delta(i) := i^{\beta-1}$ for some $\beta \in (0,1)$ that we will choose later. For this choice of β, we have $\delta(i) < \delta(j)$ and $v(i) > v(j)$ for $0 \leq j < i \leq N$. Also, for $N \to \infty$ we have $\lim_{i \to \infty} v(i) = \infty$. We call instances of the Region Choosing problem in this form *β-decreasing*. Observe that in every β-decreasing instance the optimum solution of cardinality $i \leq N$ is to take all i elements from region R_i. This solution has value $f^\star_i = i^\beta$.

In order to impose a lower bound on the best-possible competitive ratio for β-decreasing instances, we need some insights into the structure of incremental solutions with an optimal competitive ratio. First, consider a solution that picks only $i' < i$ elements from region R_i. In this case, we could have picked i' elements from region $R_{i'}$ instead – this would only improve the solution, since densities are decreasing. Secondly, if we take i elements from region R_i, it is always beneficial to take them in an uninterrupted sequence before taking any elements from a region R_j with $j > i$: Our objective depends only on the region with the

[1] This and all other missing proofs are deferred to the full version of this paper.

most value, therefore it never helps to take elements from different regions in an alternating fashion. This leads us the following observation.

▶ **Observation 10.** *For every β-decreasing instance of* Region Choosing *there is an incremental solution with optimal competitive ratio of the following structure: For $k_0 < k_1 < \cdots < k_m \in \mathbb{N}$ with $m \in \mathbb{N}$, it takes k_0 elements from region R_{k_0}, followed by k_1 elements from R_{k_1}, and so on, until finally k_m elements from region R_{k_m} are chosen.*

Thus, we can describe an algorithm for the region-choosing problem by an increasing sequence of region indices $k_0, \dots, k_m$. Note that, in order to have a bounded competitive ratio if $N \to \infty$, we must have $m \to \infty$, since $\lim_{i\to\infty} v(i) \to \infty$. We are interested in a cardinality for which an incremental solution given by $k_0, \dots, k_m$ has a bad competitive ratio. We define

$$\alpha_i := \frac{1}{k_i} \sum_{j=0}^{i} k_j \qquad \text{for all } i \in \{0, \dots, m\}\,.$$

Observe that $\alpha_i > 1$ for all $i \in \{1, \dots, m\}$. We know that the value of the optimum solution for cardinality $\alpha_i k_i$ is $v(\alpha_i k_i) = (\alpha_i k_i)^\beta$, whereas the incremental solution only achieves a value of $v(k_i) = (k_i)^\beta$. This allows us to derive the following necessary condition on the α_i-values of ρ-competitive solutions.

▶ **Observation 11.** *If an incremental solution defined by a sequence $k_0, \dots, k_m$ is ρ-competitive for some $\rho \geq 1$, we must have*

$$\rho \geq \frac{v(\alpha_i k_i)}{v(k_i)} = \left(\frac{\alpha_i k_i}{k_i}\right)^\beta = \alpha_i^\beta \quad \iff \quad \alpha_i \leq \rho^{\frac{1}{\beta}} \qquad \text{for all } i \in \{0, \dots, m\}\,. \tag{2}$$

We will exclude a certain range of values of ρ by showing that we can find a $\beta \in (0, 1)$ such that, for a sufficiently large number of regions N, necessary condition (2) is violated. We do this by showing that, for some $i^\star \in \mathbb{N}$ and some fixed $\varepsilon > 0$, we have $\alpha_{i+1} - \alpha_i > \varepsilon$ for all $i \geq i^\star$, i.e., as i goes to ∞, condition (2) must eventually be violated. The following definition relates a value of $\beta \in (0, 1)$ to a lower bound on the competitive ratio ρ for β-decreasing instances.

▶ **Definition 12.** A pair (ρ, β) with $\rho \geq 1$ and $\beta \in (0, 1)$ is *problematic* if there is $\varepsilon > 0$ such that for all $x \in (1, \rho^{1/\beta}]$ it holds that $h_{\rho,\beta}(x) < 0$, where

$$h_{\rho,\beta}(x) := (\rho^{\frac{1}{\beta}} + \varepsilon - x)^{\frac{1}{1-\beta}} - \frac{x}{x - 1 + \varepsilon}.$$

We show that problematic pairs indeed have the intended property.

▶ **Lemma 13.** *If (ρ, β) is a problematic pair, then ρ is a strict lower bound on the competitive ratio of incremental solutions for β-decreasing instances of* Region Choosing.

All that remains is to specify a problematic pair in order to obtain a lower bound via Lemma 13. It is easy to verify that $(2.18, 0.86)$ is a problematic pair. Note that the resulting bound of 2.18 can slightly be increased to larger values below 2.19.

▶ **Theorem 14.** *There is no* 2.18*-competitive incremental* Region Choosing *algorithm.*

4 The greedy algorithm for a subclass of incremental problems

In this section, we analyze the greedy algorithm that computes an incremental solution $\vec{S}$ with $\vec{S}_k = \vec{S}_{k-1} \cup \{s_k\}$, where $s_k \in \arg\max_{s \in U \setminus \vec{S}_{k-1}} f(\vec{S}_{k-1} \cup \{s_k\})$ and $\vec{S}_0 = \emptyset$. This algorithm is well-known to have competitive ratio $\frac{e}{e-1} \approx 1.58$ if the objective function f is monotone and submodular [27]. Note that every monotone and submodular function is incremental. On the other hand, in general, the greedy algorithm does not have a bounded competitive ratio for incremental problems.

▶ **Observation 15.** *The greedy algorithm has an unbounded competitive ratio for many incremental problems, e.g.,* Knapsack, Weighted Independent Set, *and* Disjoint Paths.

We will now define a subclass of incremental problems where the competitive ratio of greedy can be bounded. Observe that submodularity of a function $f\colon 2^U \to \mathbb{R}_{\geq 0}$ implies that, for every $S \neq T \subseteq U$, there exists an element $t \in T \setminus S$ with $f(S \cup \{t\}) - f(S) \geq (f(S \cup T) - f(S))/|T \setminus S|$. Accordingly, we can define the following relaxation of submodularity.

▶ **Definition 16.** We say that $f\colon 2^U \to \mathbb{R}_{\geq 0}$ is *α-augmentable* for an $\alpha > 0$, if for every $S, T \subseteq U$ with $T \setminus S \neq \emptyset$ there exists an element $t \in T \setminus S$ with

$$f(S \cup \{t\}) - f(S) \geq \frac{f(S \cup T) - \alpha f(S)}{|T|}. \tag{3}$$

Thus, if f is α-augmentable, we can improve a greedy solution if its value is more than a factor of α away from the value of an optimal solution. This definition is meaningful in the sense that it induces an interesting subclass of incremental problems.

▶ **Lemma 17.** *The objective functions of* Maximum (Weighted) (b-)Matching *and* Maximum Bridge-Flow *are 2-augmentable, but not submodular.*

We now show the first part of Theorem 4.

▶ **Theorem 18.** *If the objective function of an incremental problem is α-augmentable, the greedy algorithm is $\alpha\frac{e^\alpha}{e^\alpha-1}$-competitive.*

Proof. Let $\vec{S}_i$ be our greedy incremental solution after i elements have been added. Let us focus on an arbitrary cardinality $k > \alpha$, and say that for this cardinality we have $f_k^\star = \alpha f(\vec{S}_k) + \beta$, for some $\beta > 0$. For cardinalities $k \leq \alpha$, note that $f_1^\star = f(\vec{S}_1)$ and that $f_k^\star \leq k f_1^\star \leq \alpha f_1^\star \leq \alpha f(\vec{S}_k)$ due to sub-additivity. We will show that we must have $\beta \leq \alpha f(\vec{S}_k)/(e^\alpha - 1)$, which proves the theorem for cardinalities $k > \alpha$.

First, let us define $p_k := f(\vec{S}_k) - f(\vec{S}_{k-1})$ to be the additional value obtained by adding the k-th element to our greedy solution $\vec{S}_k$. We claim that

$$p_i \geq \frac{\beta + \alpha \sum_{j=i+1}^{k} p_j}{k - \alpha} \quad \text{for any positive integer } i < k, \text{ and} \quad p_k \geq \frac{\beta}{k - \alpha}\,. \tag{4}$$

To prove (4), we apply (3) for $S = \vec{S}_{i-1}$ and $T = S_k^\star$, which guarantees the existence of a $t \in S_k^\star \setminus \vec{S}_{i-1}$ such that

$$f(\vec{S}_{i-1} \cup \{t\}) - f(\vec{S}_{i-1}) \geq \frac{f(\vec{S}_{i-1} \cup S_k^\star) - \alpha f(\vec{S}_{i-1})}{k} \geq \frac{f_k^\star - \alpha f(\vec{S}_{i-1})}{k} \geq \frac{\beta + \alpha \sum_{j=i}^{k} p_j}{k}.$$

The last inequality holds since we assume $f_k^\star = \alpha f(\vec{S}_k) + \beta$ and for any $i \leq k$ we know that $f(\vec{S}_k) - f(\vec{S}_{i-1}) = \sum_{j=i}^{k} p_j$. Since we construct $\vec{S}_i$ greedily, we have $f(\vec{S}_i) \geq f(\vec{S}_{i-1} \cup \{t\})$

and thus $p_i \geq (\beta + \alpha \sum_{i \leq j \leq k} p_j)/k$. Rearranging to isolate p_i we get the bounds in (4). Next, we claim that

$$p_i \geq \frac{\beta}{k-\alpha} \cdot \left(\frac{k}{k-\alpha}\right)^{k-i} \quad \text{for all } i \leq k. \tag{5}$$

We prove this by induction for decreasing values of i. The induction base for $i = k$ follows directly from (4). For the induction step, we assume that the formula holds for all p_j with $j \in \{i+1, \dots, k\}$. By (4) and the inductive hypothesis, we have

$$\begin{aligned}(k-\alpha) \cdot p_i &\geq \beta + \sum_{j=i+1}^{k} \alpha p_j \geq \beta + \alpha \sum_{j=i+1}^{k} \frac{\beta}{k-\alpha} \cdot \left(\frac{k}{k-\alpha}\right)^{k-j} \\ &= \beta + \frac{\alpha\beta}{k-\alpha} \sum_{j=0}^{k-i-1} \left(\frac{k}{k-\alpha}\right)^{j} = \beta + \frac{\alpha\beta}{k-\alpha} \cdot \frac{(\frac{k}{k-\alpha})^{k-i} - 1}{\frac{k}{k-\alpha} - 1} = \beta \cdot \left(\frac{k}{k-\alpha}\right)^{k-i}, \end{aligned} \tag{6}$$

which shows that the formula also holds for p_i, and thus completes the induction step.

We are now ready to prove the theorem. Recall that we assumed $f_k^\star = \alpha f(\vec{S}_k) + \beta$ and want to show that $\beta < \alpha f(\vec{S}_k)/(e^\alpha - 1)$. We have

$$f(\vec{S}_k) = \sum_{i=1}^{k} p_i \overset{(6)}{\geq} \frac{\beta}{k-\alpha} \sum_{i=1}^{k} \left(\frac{k}{k-\alpha}\right)^{k-i} = \frac{\beta}{k-\alpha} \sum_{i=0}^{k-1} \left(\frac{k}{k-\alpha}\right)^{i} = \frac{\beta}{k-\alpha} \frac{(\frac{k}{k-\alpha})^k - 1}{\frac{k}{k-\alpha} - 1}$$

which can be rearranged to

$$\beta \leq \frac{(k-\alpha)\left(\frac{k}{k-\alpha} - 1\right) f(\vec{S}_k)}{(\frac{k}{k-\alpha})^k - 1} = \frac{\alpha f(\vec{S}_k)}{(\frac{k}{k-\alpha})^k - 1} \leq \frac{\alpha f(\vec{S}_k)}{e^\alpha - 1}$$

The last inequality holds since for any $x > 0$ we have $(1+1/x)^{x+1} \geq e$, and thus for $x = k/\alpha - 1$ it follows that $e \leq \left(1 + \frac{1}{\frac{k}{\alpha} - 1}\right)^{\frac{k}{\alpha}} = \left(1 + \frac{\alpha}{k-\alpha}\right)^{\frac{k}{\alpha}} = \left(\frac{k}{k-\alpha}\right)^{\frac{k}{\alpha}}$ and thus $(\frac{k}{k-\alpha})^k \geq e^\alpha$. ◀

4.1 Lower bound

We now show the second part of Theorem 4, i.e., we show a matching lower bound on the competitive ratio of the greedy algorithm. We show this by constructing the following family of instances for the Maximum Bridge-Flow problem. For $k \in \mathbb{N}$, we define a graph $G_k = (V_k, E_k)$ with designated nodes s and t by

$$\begin{aligned} V_k &:= \{s, t\} \cup \left\{v_i^1, v_i^4 \;\middle|\; i = 1, \dots, 2k\right\} \cup \left\{v_i^2, v_i^3 \;\middle|\; i = 1, \dots, 4k\right\}, \\ E_k &:= E_k^1 \cup E_k^\infty \cup \bigcup_{i=1}^{2k} E_{k,i} \cup \bigcup_{i=1}^{2k} E'_{k,i}, \\ E_k^1 &:= \left\{(s, v_i^2), (v_i^3, t) \;\middle|\; i = 1, \dots, k\right\}, \\ E_k^\infty &:= \left\{(s, v_{3k+i}^2), (v_i^2, v_i^3), (v_{3k+i}^2, v_{3k+i}^3), (v_i^3, t) \;\middle|\; i = 1, \dots, k\right\}, \\ E_{k,i} &:= \left\{(s, v_i^1), (v_i^1, v_{k+i}^2), (v_{k+i}^2, v_{k+i}^3), (v_{k+i}^3, v_i^4), (v_i^4, t)\right\} \quad \text{for all } i = 1, \dots, 2k, \\ E'_{k,i} &:= \left\{(v_i^1, v_j^2), (v_{3k+j}^3, v_i^4) \;\middle|\; j = 1, \dots, k\right\} \quad \text{for all } i = 1, \dots, 2k. \end{aligned}$$

The edge capacities $u_k \colon E_k \to \mathbb{R}_{\geq 0}$ are given by $u_k(e) = (\frac{k}{k-1})^{2k+1-i}$ for $e \in E_{k,i}$, by $u_k(e) = \frac{1}{k}(\frac{k}{k-1})^{2k+1-i}$ for $e \in E'_{k,i}$, by $u_k(e) = 1$ for $e \in E_k^1$, and by $u_k(e) = \infty$ for $e \in E_k^\infty$.

For every G_k, we choose a directed s-t-cut $C_k := \{(v_i^2, v_i^3) \mid i = 1, \ldots, 4k\}$. Without loss of generality, we will assume in the following that we can resolve all ties in the greedy algorithm to our preference. This can be done formally by adding some very small offsets to the edge weights, but we omit this for clarity. Now consider how the greedy algorithm operates on graph G_k.

▶ **Lemma 19.** *In step $j \in \{1, \ldots, 2k\}$, the greedy algorithm picks edge (v_{k+j}^2, v_{k+j}^3).*

With this, we are ready to show the following result, which, together with Lemma 17, implies the second part of Theorem 4.

▶ **Theorem 20.** *The greedy algorithm has competitive ratio at least $\frac{2e^2}{e^2-1} \approx 2.313$ for* Maximum Bridge-Flow.

Proof. By Lemma 19, the greedy algorithm picks the edges $(v_{k+1}^2, v_{k+1}^3), \ldots, (v_{3k}^2, v_{3k}^3)$ in the first $2k$ steps. Thus, after step $2k$, greedy can send an s-t-flow of value

$$\sum_{i=1}^{2k} \left(\frac{k}{k-1}\right)^i = \left(\frac{\left(\frac{k}{k-1}\right)^{2k+1} - 1}{\frac{k}{k-1} - 1} - 1\right) = (k-1)\left(\frac{k}{k-1}\right)^{2k+1} - k\,.$$

On the other hand, the solution of size $2k$ consisting of the edges $(v_1^2, v_1^3), \ldots, (v_k^2, v_k^3)$, and $(v_{3k+1}^2, v_{3k+1}^3), \ldots, (v_{4k}^2, v_{4k}^3)$ results in an (optimal) flow value of

$$2k + 2\sum_{i=1}^{2k} \left(\frac{k}{k-1}\right)^i = 2(k-1)\left(\frac{k}{k-1}\right)^{2k+1}.$$

This corresponds to a competitive ratio of

$$\frac{2(k-1)\left(\frac{k}{k-1}\right)^{2k+1}}{(k-1)\left(\frac{k}{k-1}\right)^{2k+1} - k} = \frac{2\left(\frac{k}{k-1}\right)^{2k}}{\left(\frac{k}{k-1}\right)^{2k} - 1} = \frac{2\left(\frac{k}{k-1}\right)^{2(k-1)+2}}{\left(\frac{k}{k-1}\right)^{2(k-1)+2} - 1}\,.$$

Substituting $x := k - 1$ and using the identity $\lim_{x\to\infty}(1 + 1/x)^x = e$, we get the lower bound on the competitive ratio of the greedy algorithm claimed in Theorem 4 in the limit:

$$\lim_{x\to\infty} \frac{2\left(\frac{x+1}{x}\right)^{2x}\left(\frac{x+1}{x}\right)^2}{2\left(\frac{x+1}{x}\right)^{2x}\left(\frac{x+1}{x}\right)^2 - 1} = \lim_{x\to\infty} \frac{2e^2\left(\frac{x+1}{x}\right)^2}{2e^2\left(\frac{x+1}{x}\right)^2 - 1} = \frac{2e^2}{e^2-1}\,.$$ ◀

5 Conclusion

We have defined a formal framework that captures a large class of incremental problems and allows for incremental solutions with bounded competitive ratio. We also defined a new and meaningful subclass consisting of problems with α-augmentable objective functions for which the greedy algorithm has a bounded competitive ratio. Hopefully our results can inspire future work on incremental problems from a perspective of competitive analysis.

Obvious extensions of our results would be to close the gap between our bounds of 2.618 and 2.18 for the best-possible competitive ratio of incremental algorithms. In particular, it would be interesting whether or not the bound of 2.313 for the greedy algorithm in the 2-augmentable setting can be beaten by some other incremental algorithm already in the general setting. Also, the α-augmentable and strictly submodular settings may allow for

better incremental algorithms than the greedy algorithm. Another question is whether abandoning the accountability condition yields an interesting class of problems. Finally, it may be possible to generalize our framework to problems with a continuously growing budget and a cost associated with each element, instead of a growing cardinality constraint.

Acknowledgements. We wish to thank Andreas Bärtschi and Daniel Graf for initial discussions and the the general idea that lead to the algorithm of Section 2. We are also grateful for the detailed comments provided by an anonymous reviewer.

References

1 Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the 26th Annual ACM Symposium on Theory of computing (STOC)*, pages 163–171, 1994.

2 A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, UK, 1998.

3 Moses Charikar, Chandra Chekuri, Tomas Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

4 Marek Chrobak, Claire Kenyon, John Noga, and Neal E. Young. Incremental medians via online bidding. *Algorithmica*, 50(4):455–478, 2007.

5 Varsha Dani and Thomas P. Hayes. Robbing the bandit: less regret in online geometric optimization against an adaptive adversary. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 937–943, 2006.

6 Sanjoy Dasgupta and Philip M. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005.

7 Yann Disser, Max Klimm, Nicole Megow, and Sebastian Stiller. Packing a Knapsack of Unknown Capacity. *SIAM Journal on Discrete Mathematics*, to appear.

8 U. Faigle, W. Kern, and G. Turan. On the performance of on-line algorithms for partition problems. *Acta Cybernetica*, 9(2):107–119, 1989.

9 A. Fiat and G. J. Woeginger, editors. *Online Algorithms: The State of the Art*. Springer, Berlin, 1998.

10 Dimitris Fotakis. Incremental algorithms for facility location and k-median. *Theoretical Computer Science*, 361(2–3):275–313, 2006. doi:10.1016/j.tcs.2006.05.015.

11 Ryo Fujita, Yusuke Kobayashi, and Kazuhisa Makino. Robust matchings and matroid intersections. *SIAM Journal on Discrete Mathematics*, 27(3):1234–1256, 2013. doi:10.1137/100808800.

12 Michel Goemans and Jon Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82(1–2):111–124, 1998.

13 C. Greg Plaxton. Approximation algorithms for hierarchical location problems. *Journal of Computer and System Sciences*, 72(3):425–443, 2006.

14 Magnús M. Halldórsson and Hadas Shachnai. Return of the boss problem: Competing online against a non-adaptive adversary. In *Proceedings of the 5th International Conference on Fun with Algorithms (FUN)*, pages 237–248, 2010.

15 Jeff Hartline and Alexa Sharp. Incremental flow. *Networks*, 50(1):77–85, 2007.

16 Refael Hassin and Shlomi Rubinstein. Robust matchings. *SIAM Journal on Discrete Mathematics*, 15(4):530–537, 2002. doi:10.1137/S0895480198332156.

17 Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

18 Naonori Kakimura and Kazuhisa Makino. Robust independence systems. *SIAM Journal on Discrete Mathematics*, 27(3):1257–1273, 2013. doi:10.1137/120899480.

19 Naonori Kakimura, Kazuhisa Makino, and Kento Seimi. Computing knapsack solutions with cardinality robustness. *Japan Journal of Industrial and Applied Mathematics*, 29(3):469–483, 2012. doi:10.1007/s13160-012-0075-z.

20 Yusuke Kobayashi and Kenjiro Takazawa. Randomized strategies for cardinality robustness in the knapsack problem. *Theoretical Computer Science*, pages 1–10, 2016. doi:10.1016/j.tcs.2016.12.019.

21 A. Krause, H. B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801, 2008.

22 Eugene L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.

23 Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM Journal on Computing*, 39(8):3633–3669, 2010. doi:10.1137/070698257.

24 Jannik Matuschke, Martin Skutella, and José A. Soto. Robust randomized matchings. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1904–1915, 2014.

25 Nicole Megow and Julian Mestre. Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science (ITCS)*, pages 495–504, 2013.

26 Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM Journal on Computing*, 32(3):816–832, 2003. doi:10.1137/S0097539701383443.

27 G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14(1):265–294, 1978.

Deterministic Partially Dynamic Single Source Shortest Paths in Weighted Graphs*

Aaron Bernstein

Technical University of Berlin, Berlin, Germany
bernstei@gmail.com

Abstract

In this paper we consider the decremental single-source shortest paths (SSSP) problem, where given a graph G and a source node s the goal is to maintain shortest distances between s and all other nodes in G under a sequence of online adversarial edge deletions. In their seminal work, Even and Shiloach [JACM 1981] presented an exact solution to the problem in unweighted graphs with only $O(mn)$ total update time over all edge deletions. Their classic algorithm was the state of the art for the decremental SSSP problem for three decades, even when approximate shortest paths are allowed.

The first improvement over the Even-Shiloach algorithm was given by Bernstein and Roditty [SODA 2011], who for the case of an unweighted and undirected graph presented a $(1+\epsilon)$-approximate algorithm with constant query time and a total update time of $O(n^{2+o(1)})$. This work triggered a series of new results, culminating in a recent breakthrough of Henzinger, Krinninger and Nanongkai [FOCS 14], who presented a $(1+\epsilon)$-approximate algorithm for undirected weighted graphs whose total update time is near linear: $O(m^{1+o(1)}\log(W))$, where W is the ratio of the heaviest to the lightest edge weight in the graph. In this paper they posed as a major open problem the question of derandomizing their result.

Until very recently, all known improvements over the Even-Shiloach algorithm were randomized and required the assumption of a non-adaptive adversary. In STOC 2016, Bernstein and Chechik showed the first *deterministic* algorithm to go beyond $O(mn)$ total update time: the algorithm is also $(1+\epsilon)$-approximate, and has total update time $\tilde{O}(n^2)$. In SODA 2017, the same authors presented an algorithm with total update time $\tilde{O}(mn^{3/4})$. However, both algorithms are restricted to undirected, unweighted graphs. We present the *first* deterministic algorithm for *weighted* undirected graphs to go beyond the $O(mn)$ bound. The total update time is $\tilde{O}(n^2\log(W))$.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases Shortest Paths, Dynamic Algorithms, Deterministic, Weighted Graph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.44

1 Introduction

The objective of dynamic graph algorithms is to handle an online sequence of update operations while maintaining a desirable functionality on the graph, e.g., the ability to answer shortest path queries. An update operation may involve a deletion or insertion of an edge or a node, or a change in an edge's weight. In case the algorithm can handle only deletions and weight increases it is called decremental, if it can handle only insertions and weight decreases it is called incremental, and if it can handle both it is called fully dynamic.

* This work was supported by the Einstein Grant at Technical University Berlin.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 44; pp. 44:1–44:14

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

In this paper we consider the problem of (approximate) single source shortest paths (SSSP) in unweighted undirected graphs, in the decremental setting. Specifically, given an undirected graph G with positive weights and a source node s, our algorithm needs to preform the following operations:

1. Delete(e) – delete the edge e from the graph
2. Increase-Weight(e) – increase the weight of e
3. Distance(v) – return the distance between s and v, i.e., $\mathbf{dist}(s, v)$, in the current graph G.

Fully dynamic shortest paths has a very clear motivation, as computing shortest paths in a graph is one of the fundamental problems of graph algorithms, and many shortest path applications must deal with a graph that is changing over time. The incremental setting is somewhat more restricted, but is applicable to any setting in which the network is only expanding. The decremental setting is often very important from a theoretical perspective, as decremental shortest paths (and decremental *single source* shortest paths especially) are used as a building block in a very large variety of fully dynamic shortest paths algorithms; see e.g. [16, 17, 3, 2, 20, 1] to name just a few. Decremental shortest paths can also have applications to non-dynamic graph problems; see e.g. Madry's paper on efficiently computing multicommodity flows [18].

We say that an algorithm has an approximation guarantee of α if its output to the query Distance(v) is never smaller than the actual shortest distance and is not more than α times the shortest distance. Dynamic algorithms are typically judged by two parameters: the time it takes the algorithm to adapt to an update (the Delete or Increase-Weight operation), and the time to process a query (the Distance operation). Typically one tries to keep the query time small (polylog or constant), while getting the update time as low as possible. All the algorithms discussed in this paper have constant query time, unless noted otherwise. In the decremental setting, which is the focus of this paper, one usually considers the aggregate sum of update times over the *entire* sequence of deletions, which is referred to as the *total update time*.

Related Work

The most naive solution to dynamic SSSP is to simply invoke a static SSSP algorithm after every deletion, which requires $\tilde{O}(m)$ time, using e.g. Dijkstra's algorithm. (The $\tilde{O}$ notation suppresses polylogarithmic factors.) For unweighted graphs, since there can be a total of m deletions, the total update time for the naive implementation is thus $\Omega(m^2)$.

For fully dynamic SSSP, nothing better than the trivial $O(m)$ time per update is known. That is, we do not know how to do better than reconstructing from scratch after every edge update. For this reason, researchers have turned to the decremental (or incremental) case in search of a better solution. The first improvement stems all the way back to 1981, when Even and Shiloach [8] showed how to achieve total update time $O(mn)$ in unweighted undirected graphs. A similar result was independently found by Dinitz [7]. This was later generalized to directed graphs by Henzinger and King [9]. This $O(mn)$ total update time bound is still the state of art, and there are conditional lower bounds [19, 15] showing that it is in fact optimal up to log factors. (The reductions are to boolean matrix multiplication and the online matrix-vector conjecture respectively).

These lower bounds motivated the study of the approximate version of this problem. In 2011, Bernstein and Roditty [6] presented the first algorithm to go beyond the $O(mn)$ bound of Even and Shiloach [8]: they presented a $(1+\epsilon)$-approximate decremental SSSP algorithm for undirected unweighted graphs with $O(n^{2+O(1/\sqrt{\log n})}) = O(n^{2+o(1)})$ total

update time. Henzinger, Krinninger and Nanongkai [13] later improved the total update time to $O(n^{1.8+o(1)} + m^{1+o(1)})$, and soon after the same authors [11] achieved a close to optimal total update time of $O(m^{1+o(1)} \log W)$ in undirected weighted graphs, where W is the largest weight in the graph (assuming the minimum edge weight is 1). The same authors also showed that one can go beyond $O(mn)$ total update time bound in *directed* graphs (with a $(1+\epsilon)$ approximation) [12, 14], although the state of art is still only a small improvement: total update time $O(mn^{0.9+o(1)} \log W)$.

However, every single one of these improvements over the $O(mn)$ bound relies on randomization, and has to make the additional assumption of a *non-adaptive* adversary. In particular, they all assume that the updates of the adversary are completely independent from the shortest paths or distances returned to the user, i.e. that the updates are fixed in advance. This makes these algorithms unsuitable for many settings, and also prevents us from using them as a black box data structure. For this reason, it is highly desirable to have deterministic algorithms for the problem.

Very recently, in STOC 2016, Bernstein and Chechik presented the first *deterministic* algorithm to go beyond $O(mn)$ total update time, again with a $(1+\epsilon)$ approximation. The total update time is $\tilde{O}(n^2)$ [4]. They followed this up with a second deterministic algorithm [5] that has total update time $\tilde{O}(n^{1.5}\sqrt{m}) = \tilde{O}(mn^{3/4})$. Both algorithms rely on the same basic technique, so this is currently the only know technique for deterministically breaking through the $O(mn)$ barrier. However, both results above were limited to undirected, unweighted graphs. In this paper, we show that this core technique can also be applied to weighted graphs.

Our Results

▶ **Theorem 1.** *Let G be an undirected graph with positive edge weights subject to a sequence of edge deletions and weight increases, let s be a fixed source, and let W be the ratio between the largest and smallest edge weights in the graph. there exists a* deterministic *algorithm that maintains $(1+\epsilon)$-approximate distances from s to every vertex in total update time $O(m \log^2(n) \log(nW) + n^2 \log(n) \log(nW)\epsilon^{-2})$. The query time is $O(1)$. (Like most decremental shortest paths algorithms, our result can very easily be extended with the same update time to the incremental case, where the update sequence contains edge insertions and weight increases.)*

(Technical Note: In weighted graphs the number of updates can be very large – i.e. if each update just increases some edge weight by some small ϵ – and so in addition to the total update time above, the algorithm necessarily requires an additional $O(1)$ per update. This $O(1)$ factor is present in all dynamic algorithms, and is typically omitted.)

Our algorithm does not match the randomized state of the art of $\tilde{O}(m^{1+o(1)} \log(W))$ [11], but it is optimal up to log factors for dense graphs, and is the first deterministic algorithm to go beyond the $O(mn)$ barrier in weighted graphs. The algorithm is also much simpler than the randomized algorithm for weighted graphs, and it does not incur the extra $n^{o(1)}$ factor present in the randomized algorithms. Thus, our algorithm is in fact faster than *all* existing randomized algorithms for the problem in dense weighted graphs where $m = \Omega(n^{2-1/\sqrt{\log(n)}})$.

As a final remark, we note that all previous deterministic algorithms to go beyond the $O(mn)$ bound [4, 5] have the strange drawback that there is no obvious way to return an approximate path, only a distance. Our algorithm unfortunately shares this drawback. The reason is that whereas in other dynamic shortest path algorithms the distance returned corresponds to some path in the graph, our distance involves an additive error that is bounded

through a structural claim about the non-existence of certain paths, but does not itself correspond to an actual path (see Lemma 4.4 in [4], Lemma 5.3 in [5], and Lemma 5 in our paper.)

Preliminaries

In our model, an undirected weighted graph is subject to deletions and weight increases. All weights in the original graph are positive. Let $G = (V, E)$ always refer to the *current* versions of the graph. Let m refer to the number of edges in the original graph, and n to the number of vertices. Let $w(u, v)$ refer to the weight of edge (u, v). *Let us assume for simplicity that the minimum weighted in the original graph is at least 1*, and note that since edge weights only increase, this will be true of all version of the graph. Let W be the largest edge weight to ever appear in the graph. For any pair of vertices u, v, let $\pi(u, v)$ be the shortest $u - v$ path in G (ties can be broken arbitrarily), and let $\mathbf{dist}(u, v)$ be the length of $\pi(u, v)$. Let s be the fixed source from which our algorithm must maintain approximate distances, and let ϵ refer to our approximation parameter; when the adversary queries the distance to a vertex v, the algorithm's guarantee is to return a distance $\mathbf{dist}'(v)$ such that $\mathbf{dist}(s, v) \leq \mathbf{dist}'(s, v) \leq (1 + O(\epsilon))\mathbf{dist}(s, v)$. Note that the $(1 + O(\epsilon))$ approximation factor can always be reduced to $(1 + \epsilon)$ without affecting the asymptotic running time by simply starting with a suitably smaller ϵ'.

Given any two sets S, T we define the set difference $S \setminus T$ to contain all elements s such that $s \in S$ but $s \notin T$. We will measure the update time of the dynamic subroutines used by our algorithm in terms of their *total* update time over the entire sequence of edge changes. Note that although edges in the main graph G are only being deleted, there may be edge insertions into the auxiliary graphs used by the algorithm.

2 High Level Overview

Our algorithm extends the techniques in the STOC 2016 paper of Bernstein and Chechik [4] from unweighted to weighted graphs. We now briefly summarize their approach for unweighted graphs, aiming in this overview for total update time $\tilde{O}(n^{2.5})$ instead of $\tilde{O}(n^2)$. We start with the well known fact that the classic Even and Shiloach algorithm has total update time $O(n^{2.5})$ [8] if we only care about distances less than $\sqrt{n}$. So the hard case is long distances in a dense graph. The key observation of [4] is that these two problematic poles cannot coexist. In particular, Bernstein and Chechik showed that any shortest path contains at most $3\sqrt{n}$ vertices of degree more than $\sqrt{n}$. The basic reason is that if we look at every third high-degree vertex on the shortest path, then these vertices must have mutually disjoint neighborhoods, since otherwise there would be a shorter path between them of length 2; the claim then follows because each high-degree vertex has at least $\sqrt{n}$ neighbors, and there are only n vertices in total. Thus, Bernstein and Chechik show that the total contribution of high-degree vertices to the shortest path is small. They then argue that this allows us to effectively "ignore" all vertices of degree $\geq \sqrt{n}$, at the cost of an additive error of only $O(\sqrt{n})$. (we do not go into details of this ignoring here.) This completed their result because for distances less than $O(\sqrt{n}\epsilon^{-1})$ the classic Even and Shiloach has total update time $O(n^{2.5}\epsilon^{-1})$, whereas for longer distances we can afford the $O(\sqrt{n})$ additive error from ignoring high degree vertices (it is subsumed into the $(1 + \epsilon)$ multiplicative error), so the resulting graph only $O(n^{1.5})$ edges.

This technique has no easy extension to weighted graphs via scaling. To see this, say there were two edge weights, 1 and $\sqrt{n}$, and that every vertex had very few incident edges

of weight 1, but $\sqrt{n}$ edges of weight $\sqrt{n}$. Then it is easy to see that distances in the graph could still be as large as n (as opposed to $\sqrt{n}$ in the unweighted case). But we cannot scale weights down, or use some sort of hop-distance technique found elsewhere in the literature, because although the heavy edges are the ones that cause the graph to be dense, the shortest path could still contain many edges of weight 1. There is thus no way in a weighted graph to ignore high-degree vertices.

To overcome this, we switch focus to a notion of degree that considers edge weights. In particular, note that in the example above, although the graph was dense, the total number of edges of weight 1 was small. The first step in our result is to formalize this observation, and show that although we cannot simply ignore vertices of high degree, for any vertex v we *can* ignore its low-weight incident edges if there are many of them. The proof of this is similar to the proof that we can afford to ignore dense vertices in unweighted graphs, but because we have to deal with many edge different weights at once, the analysis is more sophisticated.

The problem is that on its own ignoring large neighborhoods of low-weight edges doesn't help: the result is a graph that is guaranteed to only have a small number of low-weight edges (and a medium number of medium-weight edges), but the graph can still be very dense with high-weight edges. To overcome this, we develop a variation on the standard Even and Shiloach algorithm [8] which is more efficient at dealing with heavy edges but incurs a $(1+\epsilon)$ approximation. The basic intuition is as follows, though the details are somewhat complex. The classic algorithm guarantees that we only touch an edge (u, v) when the distance $\mathbf{dist}(s, u)$ or $\mathbf{dist}(s, v)$ increases. But if (u, v) has high weight, and $\mathbf{dist}(s, u)$ only increases by 1, then the distance increase is insignificant with respect to the total weight on (u, v), and so we can afford to ignore it. The number of times we touch edge (u, v) thus ends up being proportional to $\epsilon^{-1}/w(u, v)$.

All in all, our algorithm is based on the framework of Bernstein and Chechik for unweighted graphs [4], but requires significant modifications both to the high-level orientation (we need a new notion of sparsity that depends on edge weights), as well as to the technical details. The basic approach introduced in [4] of ignoring certain classes of dense vertices has proved quite versatile and powerful (see the two very different applications of it in [4] and [5]), and is currently the *only* known approach for going beyond $O(mn)$ deterministically, so our extension to weighted graphs might be useful in further applications of this approach. Also, our extension of the classic Even and Shiloach algorithm is presented in extreme generality, and might prove useful in other dynamic shortest path problems where one has a graph that has more edges of high weight than of low weight.

3 The Threshold Graph

In our algorithm, each vertex will have a weight cutoff, denoted $\mathbf{cut\text{-}off}(v)$, and will end up "ignoring" the edges below that cutoff. As described in the high-level overview above, the required degree to ignore edges goes up as the edge weights go up.

▶ **Definition 2.** Let the level of edge (u, v) be $\mathbf{level}(u, v) = \lfloor \log(w(u, v)) \rfloor$. For any vertex v, let $I(v)$ contain all edges incident to v (I for incident), and let $I_i(v)$ contain all edges incident to v of level i or less. Now, for any positive threshold τ (not necessarily integral), and any vertex v, we define $\mathbf{cut\text{-}off}_\tau(v)$ to be the *maximum* index i such that $I_i(v)$ contains at least $\tau 2^i$ edges; if no such index exists, $\mathbf{cut\text{-}off}_{\tau(v)}$ is set to 0. We say that an edge (u, v) is τ-heavy if $\mathbf{level}(u, v) \leq \mathbf{cut\text{-}off}_\tau(u)$ OR $\mathbf{level}(u, v) \leq \mathbf{cut\text{-}off}_\tau(v)$; we say that it is τ-light otherwise. Let $\textsc{heavy}(\tau)$ be the set of all τ-heavy edges in G, and let $\textsc{light}(\tau)$ be the set of all τ-light edges in G. Let $G[\textsc{heavy}(\tau)]$ be the graph $(V, \textsc{heavy}(\tau))$. Note that the levels,

cut-offs, heaviness, and lightness are *always* defined with respect to the main graph G, never with respect to auxiliary graphs. Since τ is usually clear from context, we often just write **cut-off**(v), heavy, and light.

▶ **Definition 3.** Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, and a positive threshold τ, define the threshold graph $G_\tau = (V_\tau, E_\tau)$ as follows

- V_τ contains every vertex $v \in V$.
- V_τ also contains an additional vertex c for each connected component C in the graph $G[\text{HEAVY}(\tau)] = (V, \text{HEAVY}(\tau))$
- E_τ contains all edges in $\text{LIGHT}(\tau)$, with their original weights.
- For every vertex $v \in V$, E_τ contains an edge from v to c of weight $1/2$, where c is the component vertex in $V_\tau \setminus V$ that corresponds to the component C in $G[\text{HEAVY}(\tau)]$ that contains v.

For any pair of vertices $s, t \in V$ define $\pi_\tau(s,t)$ to be the shortest path from s to t in G_τ, and define $\mathbf{dist}_\tau(s,t)$ to be the weight of this path.

▶ **Lemma 4.** *For any i, G_τ contains at most n edges of weight $1/2$ that are not in E (one per vertex in V), as well at most $O(n \cdot \tau \cdot 2^i)$ edges at level i.*

Proof. An edge $(u,v) \in E$ of level i is only in E_τ if (u,v) is τ-light, i.e. if **cut-off**$(u) < i$ AND **cut-off**$(v) < i$. But a vertex with **cut-off** less than i has by definition less than $2^i\tau$ incident edges of level i. ◀

The following Lemma is the weighted analogue of Lemma 4.4 in [4], but the most direct extension of that proof to the weighted case would analyze each of the $\log(W)$ edge levels separately, and thus multiply the error by $\log(W)$, eventually leading to a $\log^3(W)$ factor in the total running time of the algorithm. To avoid this extra error, we need a somewhat more sophisticated analysis than in Lemma 4.4 of [4]

▶ **Lemma 5.** *For any graph $G = (V, E)$, any positive integer threshold τ, and any pair of vertices $s, t \in V$:*

$$\mathbf{dist}_\tau(s,t) \leq \mathbf{dist}(s,t) < \mathbf{dist}_\tau(s,t) + \frac{14n}{\tau}.$$

Proof. It is not hard to see that we have $\mathbf{dist}_\tau(s,t) \leq \mathbf{dist}(s,t)$. Simply consider the shortest $s-t$ path $\pi(s,t) \in G$. All the edges in $\pi(s,t) \bigcap \text{LIGHT}(\tau)$ are also in G_τ. Otherwise, for any τ-heavy edge (u,v) on $\pi(s,t)$, recall that the weight $w(u,v)$ in G is always at least 1, and note that because edge (u,v) is τ-heavy, the vertices u and v must be in the same connected component in $G[\text{HEAVY}(\tau)]$, and so there is a path of length 1 from u to v in G_τ: the edge of weight $1/2$ from u to the component vertex c, and then a second edge from c to v.

We now prove that $\mathbf{dist}(s,t) \leq \mathbf{dist}_\tau(s,t) + \frac{14n}{\tau}$. Let $\pi_\tau(s,t)$ be the shortest $s-t$ path in G_τ. Consider the edge set $E^* \subseteq E$ which contains all the τ-light edges of $\pi_\tau(s,t)$ (these edges are also in E), and all τ-heavy edges in E. That is, $E^* = \text{HEAVY}(\tau) \bigcup (\text{LIGHT}(\tau) \bigcap \pi_\tau(s,t))$. Let $G^* = (V, E^*)$. We first observe that there exists an $s-t$ path in G^*. We construct this path by following $\pi_\tau(s,t)$. $\pi_\tau(s,t)$ contains τ-light edges, which are by definition in G^*, as well as subpaths of length 2 of the form $(v,c) \circ (c,w)$, where v and w are in the same connected component C in $G[\text{HEAVY}(\tau)]$. But since v and w are in the same connected component, there is a path of only heavy edges connecting them, and all heavy edges are in G^*.

Now, let $\pi^*(s,t)$ be the *shortest* $s-t$ path in G^*. Let $\mathbf{dist}^*(s,t)$ be the length of $\pi^*(s,t)$. Since G^* is a subgraph of G, we know that $\mathbf{dist}(s,t) \leq \mathbf{dist}^*(s,t)$. We now show that

$$\mathbf{dist}^*(s,t) < \mathbf{dist}_\tau(s,t) + \frac{14n}{\tau} \tag{1}$$

which will complete the proof of the lemma.

We prove Equation 1 by showing that all the heavy edges on $\pi^*(s,t)$ have total weight at most $\frac{14n}{\tau}$: since all the light edges on $\pi^*(s,t)$ are by definition also on $\pi_\tau(s,t)$, this proves the equation.

Let V_H^* be the set of vertices on $\pi^*(s,t)$ that are incident to at least one heavy edge on $\pi^*(s,t)$. Now, for any vertex $v \in V_H^*$, let $\textsc{ball}(v)$ contain v and all vertices $u \in V$, such that there is an edge (u,v) with $\mathbf{level}(u,v) \leq \mathbf{cut\text{-}off}(v)$. Note that by definition of $\mathbf{cut\text{-}off}(v)$, we have $|\textsc{ball}(v)| \geq 2^{\mathbf{cut\text{-}off}(v)}\tau$.

Now, let P be an ordering of the vertices in $\pi^*(s,t)$ in non-increasing order of $\mathbf{cut\text{-}off}$: so if $i < j$ then $\mathbf{cut\text{-}off}(P[i]) \geq \mathbf{cut\text{-}off}(P[j])$. We will now construct a set $\mathcal{IV}$ of *independent* vertices in V_H^* as follows. We start with $\mathcal{IV}$ empty. Now, go through the vertices in V_H^* according to their order in P (so in non-increasing order of cut-off), and for each v do the following: if $\textsc{ball}(v)$ is disjoint from $\textsc{ball}(u)$ for every $u \in \mathcal{IV}$, add v to $\mathcal{IV}$. Clearly if v and w are independent then $\textsc{ball}(v)$ and $\textsc{ball}(w)$ are disjoint. But then recalling that for any v, $|\textsc{ball}(v)| \geq 2^{\mathbf{cut\text{-}off}(v)}\tau$, and that the total number of vertices in the graph is n, we have

$$\sum_{v \in \mathcal{IV}} 2^{\mathbf{cut\text{-}off}(v)} \leq n/\tau \tag{2}$$

We will end up showing that each independent vertex v is "responsible" for at most $14 \cdot 2^{\mathbf{cut\text{-}off}(v)}$ weight from heavy edges on $\pi^*(s,t)$, which will complete our proof. We say that a non-independent vertex u belongs to independent vertex v if $\mathbf{cut\text{-}off}(v) \geq \mathbf{cut\text{-}off}(u)$ and $\textsc{ball}(u)$ and $\textsc{ball}(v)$ are non-disjoint. By our independence construction, every non-independent vertex u belongs to one or more independent vertices.

▶ **Claim 6.** *If v is independent, and u belongs to v, then the distance from v to u along $\pi^*(s,t)$ is at most $4 \cdot 2^{\mathbf{cut\text{-}off}(v)}$.*

Proof. Say, for contradiction, that this was not the case. Then, since u belongs to v, there must be some vertex $z \in \textsc{ball}(v) \bigcap \textsc{ball}(u)$. But by definition of $\textsc{ball}$ we have $\mathbf{level}(v,z) \leq \mathbf{cut\text{-}off}(v)$ and $\mathbf{level}(u,z) \leq \mathbf{cut\text{-}off}(u) \leq \mathbf{cut\text{-}off}(v)$, so by definition of $\mathbf{level}$ there is a path from u to v through z of weight at most $4 \cdot 2^{\mathbf{cut\text{-}off}(v)}$, which contradicts $\pi^*(s,t)$ being the shortest path in G^*. ◀

Now, given any τ-heavy edge (x,y), let the dominant endpoint of (x,y) be the endpoint that comes earlier in P: so in particular, if x is dominant then $\mathbf{cut\text{-}off}(x) \geq \mathbf{cut\text{-}off}(y)$. We say that a τ-heavy edge (x,y) belongs to independent vertex v if the dominant endpoint of (x,y) belongs to v. Note that every τ-heavy edge in $\pi^*(s,t)$ belongs to at least one independent vertex. We now show that if v is independent, then the total weight of τ-heavy edges that belong to v is at most $14 \cdot 2^{\mathbf{cut\text{-}off}(v)}$. Otherwise (for contradiction), at least half that weight would come before or after v, so w.l.o.g, let us say that there is at least $7 \cdot 2^{\mathbf{cut\text{-}off}(v)}$ weight of edges that belong to v and come after v on $\pi^*(s,t)$ – i.e., are closer to t than v is. Let us consider the τ-heavy edge (x,y) that belongs to v and is closest to t, and say that y is closer to t than x. Note that the distance from v to y along $\pi^*(s,t)$ is at least $7 \cdot 2^{\mathbf{cut\text{-}off}(v)}$, so by Claim 6, y cannot belong to v. Thus, since edge (x,y) belongs to

v, x must be the dominant endpoint and belong to v. But if x belongs to v and dominates y then $\textbf{cut-off}(y) \leq \textbf{cut-off}(x) \leq \textbf{cut-off}(v)$, and the only way edge (x, y) could be τ-heavy is if $\textbf{level}(x, y) \leq \textbf{cut-off}(v)$, so $w(x, y) \leq 2 \cdot 2^{\textbf{cut-off}(v)}$, so the distance from v to x along $\pi^*(s, t)$ is still at least $7 \cdot 2^{\textbf{cut-off}(v)} - 2 \cdot 2^{\textbf{cut-off}(v)} = 5 \cdot 2^{\textbf{cut-off}(v)}$, which again contradicts Claim 6. Thus, the total weight of heavy edges on $\pi^*(s, t)$ is at most $14 \sum_{v \in \mathcal{IV}} 2^{\textbf{cut-off}(v)}$, which combined with Equation 2 proves Equation 1, which proves the lemma. ◀

Maintaining the threshold graph G_τ as G changes is easy, because the only hard part is maintaining the connected components C in $G[\text{HEAVY}(\tau)]$, and dynamic connectivity in undirected graphs is a well-studied problem that admits deterministic $O(\log^2(n))$ update time. The details are very similar to those for unweighted graphs (see Lemmas 4.5 and 4.6 in [4]), so we omit the proofs of the following two lemmas.

▶ **Lemma 7.** *Given a graph G subject to a decremental update sequence, and a positive integer threshold τ, we can maintain the graph G_τ in* total *time $O(m \log^2(n))$. Moreover, the total number of edges of weight $1/2$ ever inserted into the graph is $O(n \log(n))$, and the total number of edges (u, v) that have $\textbf{level}(u, v) = i$ when inserted is at most $n\tau 2^i$.*

▶ **Lemma 8.** *Given a graph G subject to a decremental update sequence, a positive integer threshold τ, and a pair of vertices u, v in G, the distance $\textbf{dist}_\tau(u, v)$ in G_τ never decreases as the graph G changes.*

4 An Extension of the Even and Shiloach Algorithm

In the paper of Bernstein and Chechik for unweighted graphs [4], the threshold graph ended up being sparse, and so it was easy to efficiently compute distances within it using the standard Even and Shiloach algorithm with total update time $O(mn)$ [8]. In our paper, however, the threshold graph is only sparse with respect to low-weight edges; that is, because heaviness and lightness is defined in terms of edge weights, the threshold graph can have many edge of high weight. We now present a modification of the Even and Shiloach algorithm that can deal more efficiently with edges of high weight, which makes it perfectly suited to our threshold graph, which has mostly high-weight edges. Like the standard Even and Shiloach algorithm, our extension runs up to a certain depth bound d.

▶ **Definition 9.** Given any number d, the function $\text{BOUND}_d(x)$ is equal to x if $x \leq d$, and to ∞ otherwise.

Note that to apply to the threshold-graph G_τ, the algorithm must be able to handle insertions, as long as they do not change distances. (See Lemmas 7, 8). To this end we need the following definition:

▶ **Definition 10.** Let $G = (V, E)$ be a graph with positive weights ≥ 1, subject a dynamic update sequence of insertions, deletions, and weight-increases. In such a context, let $\mathcal{A}$ contain the set of ALL edges (u, v) to appear in G at any point during the update sequence ($\mathcal{A}$ for all): if an edge (u, v) is deleted and then inserted again, we consider this as two separate edges in $\mathcal{A}$. For every edge $(u, v) \in \mathcal{A}$, let $w_o(u, v)$ (o for original) be the weight of (u, v) when it first appears in $\mathcal{A}$ (this might be in the original graph itself), and let $\textbf{level}_o(u, v) = \lfloor \log(w_o(u, v)) \rfloor$.

▶ **Lemma 11.** *Let $G = (V, E)$ be an undirected graph with positive integer edge weights, s a fixed source, and $d \geq 1$ a depth bound. Say that G is subject to a dynamic sequence of*

edge insertions, deletions, and weight increases, with the property that distances in G never decrease as a result of an update. Then, there exists an algorithm $\mathbf{WSES}(G, s, d)$ *(stands for weight-sensitive Even and Shiloach) that maintains approximate distances* $\mathbf{dist}'(s, v)$ *with* $\mathbf{dist}(s, v) \leq \mathbf{dist}'(s, v) \leq \text{BOUND}_d(\mathbf{dist}(s, v))(1 + \epsilon)$*, and has total update time* $O(nd + \sum_{(u,v)\in\mathcal{A}} \frac{d}{\epsilon w_o(u,v)})$.

Proof. Throughout the proof of this lemma, we will often rely on the following function:

▶ **Definition 12.** Given any positive real numbers β and x, let $\text{ROUND}_\beta(x)$ be the smallest number $y > x$ that is an integer multiple of β.

Our algorithm **WSES** follows the basic procedure of the classic Even and Shiloach algorithm [8] (denoted **ES**), with a few modifications. For this reason we can describe many of the guarantees of **ES** without looking under the hood for exactly how they are implemented, since we keep exactly the same implementation. We only modify the classic algorithm in a few key points. Recall the definition of $\mathcal{A}$ and $w_o(u, v)$ from Definition 10. Since weights only increase, we always have $w(u, v) \geq w_o(u, v)$.

The **ES** algorithm maintains for each vertex v a distance label $\delta(v)$ with the property that $\mathbf{dist}(s, v) = \delta(v)$. In particular, it maintains the invariant that $\delta(s) = 0$, and that if $(u, v) \in E$, then $\delta(v) \leq \delta(u) + w(u, v)$. Our algorithm will also maintain a label $\delta(v)$ for every v, but since we only need an approximation, we guarantee a weaker invariant:

▶ **Invariant 13** (Approximation)**.** *We always have $\delta(s) = 0$. If p is the parent of v in the tree then $\delta(v) \geq \delta(p) + w(p, v)$. Also, for all $(u, v) \in E$, $\delta(v) \leq \delta(u) + w(u, v) + \epsilon w_o(u, v)$.*

Approximation Analysis: We now show that as long as the Approximation Invariant is preserved, we always have $\mathbf{dist}(s, v) \leq \delta(v) \leq (1+\epsilon)\mathbf{dist}(s, v)$. The fact that $\delta(v) \geq \mathbf{dist}(s, v)$ follows trivially for the same reason as in classic **ES**. To prove the other side, consider the shortest path $\pi(s, v)$ in G. Let the vertices in $\pi(s, v)$ be $s_0, s_1, ..., s_q = v$. We show by induction that $\delta(s_i) \leq (1+\epsilon)\mathbf{dist}(s, s_i)$. The claim is trivially true for $i = 0$ because $\delta(s) = 0$ at all times. Now, say the claim is true for s_i, and note that $\mathbf{dist}(s, s_{i+1}) = \mathbf{dist}(s, s_i) + w(s_i, s_{i+1})$. Now, by the Approximation Invariant, we have $\delta(s_{i+1}) \leq \delta(s_i) + w(s_i, s_{i+1}) + \epsilon w_o(s_i, s_{i+1}) \leq \delta(s_i) + (1 + \epsilon)w(s_i, s_{i+1})$ (the last step follows from $w(u, v) \geq w_o(u, v)$). But then by the induction hypothesis we have $\delta(s_{i+1}) \leq (1 + \epsilon)(\mathbf{dist}(s, s_i) + w(s_i, s_{i+1})) = (1 + \epsilon)\mathbf{dist}(s, s_{i+1})$.

We must now show how to efficiently maintain the Approximation Invariant. The classic **ES** algorithm does $O(1)$ time per update for basic setup, and otherwise all the work comes from charging constant time operations to various edges (u, v) and vertices v. The update time invariant in classic **ES** is as follows: we only charge vertex v when $\delta(v)$ increases, and we only charge edge (u, v) when $\delta(u)$ or $\delta(v)$ increases. Since weights are positive integers, every $\delta(u)$ must increase by at least 1, and since we only go up to distance d, each $\delta(v)$ increases at most d times, which yields $O(md)$. To improve upon this, our algorithm maintains a slightly different update time invariant

▶ **Invariant 14** (Update-Time)**.** *A vertex v is only charged when $\delta(v)$ increases. An edge (u, v) is only charged when either* $\text{ROUND}_{\epsilon w_o(u,v)}(\delta(u))$ *or* $\text{ROUND}_{\epsilon w_o(u,v)}(\delta(v))$ *increases.*

Since for any β, $\text{ROUND}_\beta(\delta(v))$ can increase at most d/β before it exceeds d, it is clear that the Update-Time Invariant guarantees the total update bound of the lemma.

Maintaining the Two Invariants: Now, let us recall how the standard **ES** algorithm works. This algorithm only handles deletions, so that is what we focus on first: we later show how to extend our algorithm to handle weight-increases and insertions that do not change distances. We will emphasize the parts of classic **ES** that we will later change in our modified algorithm. First off, each vertex v always maintains label information about its neighbors: *so in particular, for each edge (u, v), vertex v stores a copy $\delta_v(u) = \delta(u)$* (FIRST-DIFFERENCE). The algorithm maintains a shortest path tree T from the root s. We say an edge (u, v) is a tree edge if $(u, v) \in T$, and a non-tree edge if $(u, v) \in E \setminus T$. Whenever a non-tree edge is deleted, shortest distances do not change, so the algorithm does not have to do much. Now, consider the deletion of a tree edge (u, v) where u is the parent of v in T. The algorithm checks in constant time, by cleverly storing local label information $\delta_v(z)$, if v has another edge to a vertex z with $\delta(v) = \delta_v(z) + w(z, v)$. If yes, v can be reconnected without a label increase, the edge (z, v) is added to the tree and all distance labels remain the same so we are done. If not, label $\delta(v)$ must increase, which in turn affects of the children of v, so *The algorithm then examines all the children of v and checks if they can be attached to the tree without increasing their distance in a similar manner.* (SECOND-DIFFERENCE) Some vertices will fail to be reattached with the same label, so the algorithm keeps a list of all vertices that need to be re-attached at a higher label. Each time the algorithm discovers that a label $\delta(v)$ must increase, it increases it by 1 (which is the most optimistic new label it can get) and returns it to the needs-reattachment list in at attempt to reattach v with this higher label. The algorithm examines the vertices in the list in an increasing order of their label. Note that the label of a vertex may be increased many times as a result of an update. *Whenever the label $\delta(v)$ of a vertex v changes, the algorithm must adjust $\delta_u(v)$ for every edge (u, v) in the graph.* (THIRD-DIFFERENCE.)

Note that the algorithm above fails to satisfy the Update-Time Invariant, because although a vertex v is only charged when $\delta(v)$ increases (otherwise the search stops at v and can be charged to the edge that led to v), edge (u, v) is charged whenever $\delta(u)$ or $\delta(v)$ increase even just by 1. In particular, the invariant is not preserved because of the work in SECOND-DIFFERENCE and THIRD-DIFFERENCE. The changes we implement are simple. We start with FIRST-DIFFERENCE: each vertex v will still store local label information $\delta_v(u)$, but now it might be slightly inaccurate. In particular, we always have

▶ **Invariant 15** (Local-Information). $\delta(u) \leq \delta_v(u) \leq \text{ROUND}_{\epsilon w_o(u,v)}(\delta(u))$.

A vertex v will now choose its parent based on the slightly inaccurate local labels $\delta_u(v)$ instead of the true label $\delta(u)$. Let $p(v)$ be the parent of v in the tree.

▶ **Invariant 16** (Parent-Choice). $p(v) = argmax_{u \in \text{NEIGHBORS}(v)}\{\delta_v(u) + w(u, v)\}$ *and* $\delta(v) = \delta_v(p(v)) + w(p(v), v)$.

The Parent-Choice and Local-Information invariants combined clearly guarantee the Approximation Invariant. We must now show how to maintain these efficiently. This leads us to THIRD-DIFFERENCE: when $\delta(v)$ changes, we only update $\delta_u(v)$ for every vertex u when $\text{ROUND}_{\epsilon w_o(u,v)}(\delta(v))$ increases. This ensures that we satisfy the Update-Time Invariant for this step of the algorithm, while still ensuring that all local information adheres to the Local-Information Invariant. We omit the full details, but it very easy to maintain a data structure for each vertex v, that returns in $O(1)$ time per edge all edges (u, v) for which $\text{ROUND}_{\epsilon w_o(u,v)}(\delta(v))$ increased. This is because $w_o(u, v)$ is completely fixed for each edge, while $\delta(v)$ only rises from 1 to d, so we only need d slots per vertex (slot i corresponding to $\delta(v)$ going up from i to $i + 1$), and $O(1)$ time per edge affected in each slot.

The last violation of the Update-Time Invariant was in SECOND-DIFFERENCE. When $\delta(v)$ changes, instead of processing all children u, we only need to process those for which $\delta_u(v)$ changes. But in our modified algorithm this only occurs when $\text{ROUND}_{\epsilon w_o(u,v)}(\delta(v))$ increases, in keeping with the Update-Time Invariant.

We have thus shown how to modify classical **ES** in a way that maintains the Approximation Invariant and the Update-Time Invariant while processing deletions. Increase-Weight(u, v) is processed in the same way: if (u, v) was not a tree edge we do nothing, and otherwise if u was the parent of v, then some invariants will be violated for v, and we fix these in exactly the same way as for a deletion.

We now turn to processing insertions. In classical **ES**, it is trivial to process an insertion of (u, v) that does decrease distances because the distance labels do not change, so we just spend $O(1)$ time fixing update $\delta_v(u)$ and $\delta_u(v)$. In our algorithm, however, a strange difficulty arises: because our labels are approximate, even if the insertion of (u, v) does not decrease distances, it might nonetheless violate the Approximation Invariant: say that $\mathbf{dist}(u) = \mathbf{dist}(v) = 1000$ but $\delta(v) = 1000(1 + \epsilon)$ while $\delta(u) = 1000$ and we insert and edge of weight 1. In this case we cannot afford to decrease $\delta(v)$ because our update time analysis relied on non-decreasing labels. But note that although v now violates the Approximation Invariant, $\delta(v)$ is still a $(1+\epsilon)$ approximation to $\mathbf{dist}(s, v)$, because $\mathbf{dist}(s, v)$ never decreases. We thus rely on the idea used in the "Monotone" **ES** tree of Henzinger et al. [10]: we simply never decrease distance labels. More formally, our distance labels will satisfy the following *relaxed* Approximation Invariant: after every update, for every vertex v, either $\delta(v)$ satisfies the standard Approximation Invariant, or $\delta(v) = \delta^{\text{OLD}}(v)$, where $\delta^{\text{OLD}}(v)$ was the distance label before the update. By an induction on time, it is easy to see that with this relaxed Approximation Invariant, we still always have that $\delta(v) \leq (1 + \epsilon)\mathbf{dist}(s, v)$. If $\delta(v)$ satisfies the approximation invariant, the proof is the same as in the Approximation Analysis above. Otherwise, we have $\delta(v) = \delta^{\text{OLD}}(v)$, in which case by our time induction we have $\delta^{\text{OLD}}(v) \leq (1 + \epsilon)\mathbf{dist}^{\text{OLD}}(s, v)$, and since our updates are guaranteed not to decrease distances, we have $\mathbf{dist}^{\text{OLD}}(s, v) \leq \mathbf{dist}(s, v)$ and we are done. ◀

5 The decremental SSSP algorithm

We now put together all our ingredients to proving the main Theorem 1.

▶ **Lemma 17.** *For any threshold τ, and depth bound $d = n\epsilon^{-1}\tau^{-1}$, we can maintain approximate distances $\mathbf{dist}'_\tau(s, v)$ in total time $O(m \log^2(n) + n^2 \log(n)\epsilon^{-2})$, with the property that $\mathbf{dist}'_\tau(s, v) \leq (1 + \epsilon)\text{BOUND}_d(\mathbf{dist}_\tau(s, v)) + d\epsilon$*

Proof. First we use Lemma 7 to maintain the threshold graph G_τ in $O(m \log^2(n))$ total update time. Then, setting $\beta = \frac{d\epsilon}{2n}$, for each edge $(u, v) \in G_\tau$, we round the edge weight $w(u, v)$ up to the nearest multiple of β. It is easy to see that since every shortest path in the graph contains at most n edges, this weight change incurs an additive error of at most $n\beta = d\epsilon/2$. Since all edge weights are now multiple of β, we can divide all edge weights by factor of β without changing shortest paths, which results in a scaled graph G^s_τ with integral weights for which we have to maintain shortest distances up to depth $d^s = d/\beta = 2n\epsilon^{-1}$. We now run $\mathbf{WSES}(G^s_\tau, s, d^s)$ in the scaled graph, and then scale the resulting distances back up by β. Since **WSES** is a $(1 + \epsilon)$ approximation, we get: $\mathbf{dist}'_\tau(s, v) \leq (1 + \epsilon)\text{BOUND}_d(\mathbf{dist}_\tau(s, v) + d\epsilon/2) \leq (1 + \epsilon)\text{BOUND}_d\mathbf{dist}_\tau(s, v) + d\epsilon$.

Recall that the total update time of $\mathbf{WSES}(G^s_\tau, s, d^s)$ is $O(nd^s + \sum_{(u,v)\in\mathcal{A}} \frac{d^s}{\epsilon w_o(u,v)})$. We know that $O(nd^s) = O(n^2\epsilon^{-1})$. Now, G^s_τ contains two types of edges: edges between a vertex

$v \in V$ and a component vertex $c \in V_\tau \setminus C$, and τ-light edges from the original graph. By Lemma 7, the total number of component edges ever inserted is $O(n \log(n))$, and because of our scaling they all have weight at least 1, so their total contribution to the sum is at most $n \log(n)\epsilon^{-1} \cdot d^s = O(n^2 \log(n)\epsilon^{-2})$. For edges of the original graph, let us look at the contribution of all edges (u, v) with $\textbf{level}_o(u, v) = i$. By Lemmas 4 and 7, the total number of such edges is at most $O(n\tau 2^i)$. Each such edge has $w_o(u, v) \geq 2^i$, which has been scaled down by β in G^s_τ, and so contributes $\frac{d^s \beta}{\epsilon 2^i} = \frac{d}{\epsilon 2^i}$ to the sum. Thus in total we have that each level i contributes a total of $O(n\tau d\epsilon^{-1}) = O(n^2\epsilon^{-2})$.

We complete the proof by arguing that there are only $\log(n)$ contributing levels. For simplicity, let us look at the graph G_τ before scaling. First off, edges of weight more than $d = n\epsilon^{-1}/\tau$ can be ignored, since we do not care about distances greater than d (we are working with BOUND_d). Secondly, from the definition of $\textbf{cut-off}(v)$, an edge (u, v) of weight less than $1/(2\tau)$ will always be heavy and so never appear in G_τ: if $2\tau w(u, v) \leq 2^{\textbf{level}(u,v)}\tau < 1$, then $\textbf{cut-off}(u)$ and $\textbf{cut-off}(v)$ are always at least as large as $\textbf{level}(u, v)$, so (u, v) will remain heavy. Thus, the only edges that appear in our graph have weight between $1/(2\tau)$ and $n/(\epsilon\tau)$, which corresponds to $\log(n/\epsilon) = O(\log(n))$ different levels of edge weights. ◀

▶ **Lemma 18.** *For any distance bound d, with total update time $O(m \log^2(n) + n^2 \log(n)\epsilon^{-2})$ we can maintain approximate distances $\textbf{dist}_d(s, v)$ to all vertices v such that $\textbf{dist}(s, v) \leq \textbf{dist}_d(s, v) \leq \text{BOUND}_d(\textbf{dist}(s, v)) + 15\epsilon d$*

Proof. Set $\tau = n/(\epsilon d)$. By Lemma 17, in total update time $O(m \log^2(n) + n^2 \log(n)\epsilon^{-2})$ we can maintain values $\textbf{dist}'_\tau(s, v)$ with $\textbf{dist}_\tau(s, v) \leq \textbf{dist}'_\tau(s, v) \leq (1+\epsilon)\text{BOUND}_d(\textbf{dist}_\tau(s, v)) + \epsilon d$. Now, By Lemma 5 we have $\textbf{dist}(s, v) - 14n/\tau \leq \textbf{dist}_\tau(s, v) \leq \textbf{dist}(s, v)$. Plugging in our value $\tau = n/(\epsilon d)$ we get $\textbf{dist}(s, v) - 14\epsilon d \leq \textbf{dist}_\tau(s, v) \leq \textbf{dist}(s, v)$. Thus, we have $\textbf{dist}(s, v) - 14\epsilon d \leq \textbf{dist}'_\tau(s, v) \leq \text{BOUND}_d(\textbf{dist}(s, v)) + \epsilon d$. If we return $\textbf{dist}_d(s, v) = \textbf{dist}'_\tau(s, v) + 14\epsilon d$, we get the bound in the lemma. ◀

It is now easy to prove our main Theorem 1. Observe that the maximum distance we could possibly see is nW. Thus, for each $i = 0, 1, 2, 3...\lceil \log(nW) \rceil$, we use Lemma 18 to maintain $\delta_i(v) = \textbf{dist}_{2^i}(s, v)$. By Lemma 18, the total update time is the desired $O(m \log^2(n) \log(nW) + n^2 \log(n) \log(nW)\epsilon^{-2})$. We then output as our final answer: $\textbf{dist}'(v) = \min_i\{\delta_i(v)\}$. Since for each i we have $\delta_i \geq \textbf{dist}(v)$, we have $\textbf{dist}'(v) \geq \textbf{dist}(s, v)$. We now need to show that $\textbf{dist}'(v) \leq (1 + O(\epsilon))\textbf{dist}(s, v)$. It is not hard to see that for $i = \lceil \log(\textbf{dist}(s, v)) \rceil$, we end up with $\delta_i(v) \leq \textbf{dist}(s, v) + 30\epsilon\textbf{dist}(s, v)$, as desired.

6 Conclusions

In this paper we presented the first *deterministic* decremental SSSP algorithm for *weighted* undirected graphs that goes beyond the Even-Shiloach bound of $O(mn)$ total update time. Previously such a result was only known for unweighted undirected graphs. The two main open questions are further improving this total update time, and going beyond $O(mn)$ deterministically for *directed* graphs. Finally, we recall from the introduction that all existing deterministic $o(mn)$ results including ours are only able to return approximate shortest distances, not the paths themselves.

References

1 Ittai Abraham, Shiri Chechik, and Kunal Talwar. Fully dynamic all-pairs shortest paths: Breaking the o(n) barrier. In *Approximation, Randomization, and Combinatorial Optimiz-*

ation. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain, pages 1–16, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.1.

2 Surender Baswana, Ramesh Hariharan, and Sandeep Sen. Maintaining all-pairs approximate shortest paths under deletion of edges. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 394–403. Society for Industrial and Applied Mathematics, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644171.

3 Aaron Bernstein. Fully dynamic (2 + epsilon) approximate all-pairs shortest paths with fast query and close to linear update time. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pages 693–702, 2009. doi:10.1109/FOCS.2009.16.

4 Aaron Bernstein and Shiri Chechik. Deterministic decremental single source shortest paths: beyond the o(mn) bound. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 389–397, 2016. doi:10.1145/2897518.2897521.

5 Aaron Bernstein and Shiri Chechik. Deterministic partially dynamic single source shortest paths for sparse graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 453–469, 2017. doi:10.1137/1.9781611974782.29.

6 Aaron Bernstein and Liam Roditty. Improved dynamic algorithms for maintaining approximate shortest paths under deletions. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 1355–1365, 2011. doi:10.1137/1.9781611973082.104.

7 Yefim Dinitz. Dinitz' algorithm: The original version and even's version. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, pages 218–240, 2006. doi:10.1007/11685654_10.

8 Shimon Even and Yossi Shiloach. An on-line edge-deletion problem. *Journal of the ACM*, 28(1):1–4, 1981. doi:10.1145/322234.322235.

9 Monika Henzinger and Valerie King. Fully dynamic biconnectivity and transitive closure. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 664–672, 1995. doi:10.1109/SFCS.1995.492668.

10 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Dynamic approximate all-pairs shortest paths: Breaking the o(mn) barrier and derandomization. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, FOCS, pages 538–547, 2013. doi:10.1109/FOCS.2013.64.

11 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental single-source shortest paths on undirected graphs in near-linear total update time. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science*, FOCS, pages 146–155, 2014. doi:10.1109/FOCS.2014.24.

12 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Sublinear-time decremental algorithms for single-source reachability and shortest paths on directed graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 674–683, 2014. doi:10.1145/2591796.2591869.

13 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A subquadratic-time algorithm for decremental single-source shortest paths. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 1053–1072, 2014. URL: http://dl.acm.org/citation.cfm?id=2634074.2634153.

14 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Improved algorithms for decremental single-source reachability on directed graphs. In *Proceedings of the 42nd International Colloquium, ICALP*, pages 725–736, 2015. doi:10.1007/978-3-662-47672-7_59.

15 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, pages 21–30, 2015. doi:10.1145/2746539.2746609.

16 Monika Rauch Henzinger and Valerie King. Maintaining minimum spanning forests in dynamic graphs. *SIAM J. Comput.*, 31(2):364–374, 2001. doi:10.1137/S0097539797327209.

17 Valerie King. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS, pages 81–91, 1999. URL: http://dl.acm.org/citation.cfm?id=795665.796487.

18 Aleksander Madry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 121–130, 2010. doi:10.1145/1806689.1806708.

19 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011. doi:10.1007/s00453-010-9401-5.

20 Liam Roditty and Uri Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. *SIAM J. Comput.*, 41(3):670–683, 2012. doi:10.1137/090776573.

Testing Core Membership in Public Goods Economies*

Greg Bodwin

MIT EECS, Cambridge, MA, USA
greg.bodwin@gmail.com

Abstract

This paper develops a recent line of economic theory seeking to understand public goods economies using methods of topological analysis. Our first main result is a very clean characterization of the economy's core (the standard solution concept in public goods). Specifically, we prove that a point is in the core iff it is Pareto efficient, individually rational, and the set of points it dominates is path connected.

While this structural theorem has a few interesting implications in economic theory, the main focus of the second part of this paper is on a particular algorithmic application that demonstrates its utility. Since the 1960s, economists have looked for an efficient computational process that decides whether or not a given point is in the core. All known algorithms so far run in exponential time (except in some artificially restricted settings). By heavily exploiting our new structure, we propose a new algorithm for testing core membership whose computational bottleneck is the solution of $O(n)$ convex optimization problems on the utility function governing the economy. It is fairly natural to assume that convex optimization should be feasible, as it is needed even for very basic economic computational tasks such as testing Pareto efficiency. Nevertheless, even without this assumption, our work implies for the first time that core membership can be efficiently tested on (e.g.) utility functions that admit "nice" analytic expressions, or that appropriately defined ε-approximate versions of the problem are tractable (by using modern black-box ε-approximate convex optimization algorithms).

1998 ACM Subject Classification J.4 Social and Behavioral Sciences

Keywords and phrases Algorithmic Game Theory, Economics, Algorithms, Public Goods, Coalitional Stability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.45

1 Introduction

1.1 Background on Public Goods Economics

A basic question in economics is to understand the forces governing the production of public goods. A good is *public* if its use by one person does not reduce its availability to others, and if none are excluded from using the good. Examples include public parks, research information, a clean environment, national defense, radio broadcasts, and so on.

Public goods economies were first explicitly abstracted in a classic paper by Samuelson in 1954 [24], and have since become central objects of study for economists. An important feature of public goods economies is that they are not well modeled by the individualistic "best-response dynamics" which govern familiar economic equilibrium concepts such as the Nash or Walrasian equilibrium. Rather, public goods typically arise as the result of a

* A full version of the paper is available at https://arxiv.org/abs/1705.01570.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 45; pp. 45:1–45:14


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

communal process – e.g. negotiations, treaties, or taxes – that allow the cost of production to be amortized over all agents that stand to benefit from the good. Accordingly, public goods economics inspired the development of *cooperative game theory*, which seeks to understand these cooperative dynamics and when an agreement to produce public goods is "stable" or when it is doomed to fall apart.

What, exactly, does "stability" mean in this context? The most standard notion is *coalitional stability*, which is given as follows:

▶ **Definition** (Informal)**.** Let $\mathbf{a}$ be an outcome in a public goods economy with agents N (i.e. $\mathbf{a}$ describes the amount of work each agent contributes to produce a public good). We say that $\mathbf{a}'$ is a *deviation* on $\mathbf{a}$ for a nonempty coalition of agents $C \subseteq N$ if no agents in $N \setminus C$ perform work (i.e. $a'_i = 0$ for all $i \notin C$) and all agents in C prefer $\mathbf{a}'$ to $\mathbf{a}$. If there is no deviation on $\mathbf{a}$ for any coalition, then we say $\mathbf{a}$ is *coalitionally stable*. The set of coalitionally stable outcomes is called the *core* of the economy.

Since its inception in the late 1800's [10],[1] the core and cooperative game theory in general have played major roles in many successful economic research programs. More background can be found in most modern game theory textbooks, e.g. [23, 8].

1.2 (Non-)Algorithmic Properties of Public Goods Economies

An inherent conceptual drawback of coalitional stability is its exponential-size definition. In other words, for an outcome $\mathbf{a}$ to be coalitionally stable, *every single one* of the $2^n - 1$ possible coalitions of agents must not have a deviation. Hence, the naive algorithms for testing the coalitional stability of $\mathbf{a}$ must perform computations for all $2^n - 1$ coalitions, and so they suffer exponential runtime. Coalitional stability is not a very convincing solution concept if its implicit notion of "instability" assumes that agents can quickly make exponential time computations in order to find deviations whenever they exist.

This problem has led economists studying public goods to seek more clever methods for solving computational problems related to the core of a public goods economy, which avoid this exponential behavior. Some of the initial work in this vein attacked the closely-related problem of simply outputting any core outcome. The first such solution appeared in the 1960s, when Scarf [25] proved that "balanced" games have nonempty cores, by means of an (exponential time) algorithm that outputs a core point and provably always terminates. Similar results were proved in the setting of public goods economies by Chander and Tulkens [5] and Elliot and Golub [12]. Meanwhile, followup work has suggested that the slow runtime of Scarf's algorithm may be inherent to the problem: Kintali et al. [20] showed that Scarf's algorithm cannot be improved to polynomial runtime (unless P = PPAD), and Deng and Papadimitriou [9] showed that it is NP Complete just to detect whether or not the core is empty (let alone find a point in the core), even in the simple class of graphical games, and even when Scarf's assumption of "balance" is dropped. Other notable hardness results in this vein have come from Conitzer and Sandholm [7] and Greco et al. [18].

There has also been considerable prior work on the "membership testing" problem of determining whether a point taken on input is in the core (this is the question addressed in this paper). Deng and Papadimitriou's work [9] also implies that membership testing is NP hard even in the restricted setting of graphical games, although there are straightforward

[1] Some of the early work on cooperative game theory used the name *contract curve* instead of core. The term *core* was coined in [15].

efficient algorithms in the further restricted setting where the game is superadditive. Conitzer and Sandholm [6] showed that membership testing is co-NP complete in games where coalition values have a "multiple-issue" representation in polynomial space. Faigle et al. [13] showed that the problem is NP complete in a variant of graphical games where payoffs are given by minimum spanning trees of subgraphs. Sung and Dimitrov [26] showed co-NP completeness for membership testing in "hedonic coalition formation games." Goemans and Skutella showed NP completeness for both emptiness and membership testing in "facility location games," and gave formulations of these problems as LP relaxations [16]. There has been work on games defined by *marginal contribution nets* (MC-nets) [19, 11], in which values attainable by coalitions are determined by succinct logical formulae. Li and Conitzer [22] studied emptiness testing and membership testing under various classes of formulae, and obtained various algorithms or NP hardness results depending on the complexity of the formulae allowed.

1.3 Our Results

A recently popular trend in public goods research has been to model economies as networks, and then seek to analyze the economy by studying the topological properties of the underlying "benefits network," describing the ability of agents to transfer utility to each other at any given point (see for example [3, 4, 1, 2, 12]). Much of the initial work focused on Nash equilibria [3, 4, 1, 2] of the economy. A major stride was recently taken by Elliot and Golub [12], who extended the theory to show that the Lindahl equilibria[2] of an economy are precisely the points that are eigenvalues of their own benefits network. They further discuss connections between the Lindahl equilibria and the core – in particular, in any standard model (including theirs) all Lindahl equilibria are also core outcomes [14]. However, they raise an interesting open question to more precisely characterize the core [17] in a similar vein.

Our first main result achieves this goal. We show that the core can be characterized as follows:

▶ **Theorem 1.** *Let* $\mathbf{a}$ *be an outcome in a public goods economy and let* $\mathbf{D_a}$ *be the set of points that no agent prefers to* $\mathbf{a}$*. Then* $\mathbf{a}$ *is in the core if and only if it is Pareto efficient, individually rational, and* $\mathbf{D_a}$ *is path connected.*

(Here, the definition of path connectedness is the standard topological one: for any two points $\mathbf{x}, \mathbf{y} \in \mathbf{D_a}$, there is a continuous function $f : [0, 1] \to \mathbb{R}^n_{\geq \mathbf{0}}$ satisfying $f(0) = \mathbf{x}, f(1) = \mathbf{y}$, and $f(\lambda) \in \mathbf{D_a}$ for all $0 \leq \lambda \leq 1$. The image of $[0, 1]$ under f is called a path.)

An interesting consequence of this theorem is a precise description of the relationship between Lindahl equilibria and core outcomes:

▶ **Theorem 2.** *Assuming that the utility function* $\mathbf{u}$ *is differentiable, the Lindahl equilibria of a public goods economy are precisely the core points whose core membership can be certified using only local information.*

The proof of this corollary is essentially immediate by combining Theorem 1 with a more technical phrasing of Elliot and Golub's result.

While we believe that these two structural theorems hold intrinsic interest, the second half of this paper is intended to demonstrate their power by an application to the algorithmic

[2] The Lindahl equilibria of an economy are the competitive equilibria that would be reached if market externalities were truthfully reported and then bought and sold on an open market. A formal definition is not necessary to read this paper, but can be found in e.g. [23, 8].

problem discussed earlier. We have previously suggested the intuition that the algorithmic core membership testing problem is hard because the naive algorithms must check an exponential number of coalitions for a potential deviation. However, Theorem 1 lets us avoid this brute-force behavior: after checking for Pareto efficiency and individual rationality (which is quite easy), we are left only with the task of checking whether or not $\mathbf{D_a}$ is path-connected. The complexity of this task is non-obvious, but we show that it can be done fairly efficiently, yielding the following result:

▶ **Theorem 3.** *Given an outcome* $\mathbf{a}$ *in an* n*-agent public goods economy, there is an algorithm that decides whether or not* $\mathbf{a}$ *is in the core of a public goods economy. The computational bottleneck in this algorithm is the solution of* $O(n)$ *convex programming problems on the utility function of the economy.*

Hence we essentially have the first polynomial-time tester for coalitional stability in an unrestricted public goods economy, up to the implementation of the necessary convex programming oracle. It is fairly natural in our economic metaphor to assume that convex programming should be tractable: it corresponds to the negotiation process of a group of agents trying to determine how well they can maximize a joint utility function as a group. If even this is impossible, then it is essentially hopeless to efficiently test core membership. One cannot even test the more basic property of Pareto efficiency, a necessary step towards testing core membership, without assuming some computational power along these lines.

Even so, if one does not wish to introduce such assumptions, Theorem 3 implies that several broad special cases of membership testing have efficient algorithms. The most obvious of these is when the utility function and its derivative can be described by a "nice" analytic function on which the standard derivative-based method for exact convex optimization goes through. Less obviously, if convex optimization really is hard for the given utility function (or the utility function of the economy is unknown), one can employ modern ε-approximate convex optimization solvers, which treat the utility function as a black box that can be queried, to solve certain natural ε-approximate relaxations of the core membership testing problem. We discuss this point in the conclusion of the paper, since it is easier to be specific here once the economic model is familiar.

We consider it somewhat surprising that these algorithmic results are possible, given a general dearth of positive results in the area. Moreover, the approach taken by the algorithm is fairly intuitive and seems to plausibly reflect practical behavior. Starting with the grand coalition, we show (via Theorem 1) that we can either determine that the current coalition has a deviation, or we can identify a "least-valuable player" who is formally the least likely agent to participate in a deviation. We then kill this agent and repeat the analysis on the survivors. After n rounds, we have either killed every agent (and thus determined that the given point is coalitionally stable), or we have explicitly found a surviving coalition with a deviation. It is quite reasonable to imagine that a practical search for a deviating coalition might employ a "greedy" method of iteratively killing the agent who seems to be least pulling their weight at the current agreement; an insight of Theorem 3 is that this search heuristic is in fact thorough and will provably produce the right answer.

1.4 Comparison with Prior Work

Elliot and Golub [12] recently studied the Lindahl equilibria in public goods economies, with a focus on characterizing the set of solutions rather than algorithmically computing/testing them. More specifically, they frame the typical model of public goods economies in the language of networks, and use this to equate the eigenvectors of the "benefits network" with

the Lindahl equilibria of the economy. A less general version of this networks interpretation was implicitly used in several other papers concerning Nash equilibria of public goods economies, for example [3, 4, 1, 2]. In this paper, we will adopt the more general networks-based phrasing of public goods economies used by Elliot and Golub, and we will rely on this insight in a critical way to prove our main results.

Per the discussion above, there has been lots of prior work on the algorithmic properties of the core, largely intended to confirm/refute the bounded rationality argument in some economic model. Three questions are commonly studied:

- The *membership testing* problem (discussed above): is a given outcome in the core of the game?
- The *emptiness testing* problem: is the core empty?
- The *member finding* problem: output any solution in the core of the game (if nonempty).

We remark that the latter two problems are already closed in public goods economies: Elliot and Golub [12] show that the core is never empty except in certain degenerate cases, and it can be seen from the model below that the member finding problem is essentially identical to the general problem of convex optimization (which is well beyond the scope of this economically-minded research program). Hence, this work is entirely focused on the membership testing problem.

In order to frame these three questions as proper computational problems, past work has commonly defined a "compressed" cooperative game that allows the payoffs achievable by all 2^n possible coalitions to be expressed on only $\mathsf{poly}(n)$ input bits. For example, in a seminal paper by Deng and Papadimitriou introducing this line of research [9], the authors studied *graphical games* in which weighted edges are placed between agents and the value attainable by a coalition is equal to the total weight contained in its induced subgraph. Upper and lower bounds are often obtained for these problems by exploiting particular features of the compression scheme. By contrast, our goal is to assume as little structure for the problem as possible (since our main results are upper bounds, this is the more general approach). Thus, we allow the economy to be governed by an arbitrarily complex utility function, which does not need to have a succinct representation, or even any algorithmic representation at all. Instead, we allow ourselves black-box constant-time query access to the utility function, which acts as an oracle and thus may have arbitrary complexity. The goal in this substantial generalization is to ensure that our results reveal structure of the core itself, rather than the nature of an assumed compression.

2 The Model and Basic Definitions

2.1 Notation Conventions

Given vectors $\mathbf{a}, \mathbf{a}' \in \mathbb{R}^n$, we will use the following (partial) ordering operations:

- $\mathbf{a} \geq \mathbf{a}'$ means that $a_i \geq a'_i$ for all $1 \leq i \leq n$,
- $\mathbf{a} > \mathbf{a}'$ means that $a_i > a'_i$ for all $i \leq i \leq n$, and
- $\mathbf{a} \gneq \mathbf{a}'$ means that $a_i \geq a'_i$ for all $1 \leq i \leq n$, and $a_j > a'_j$ for some $1 \leq j \leq n$.

Given a subset $C \subseteq \{1, \ldots, n\}$ and a vector $\mathbf{v} \in \mathbb{R}^n$, we write $\mathbf{v}_C$ to denote the restriction of $\mathbf{v}$ to the indices in C; that is, $\mathbf{v}_C$ is the $|C|$ length vector built by deleting the entry v_i from $\mathbf{v}$ for each $i \notin C$.

We use $\mathbf{0}, \mathbf{1}$ as shorthands for the vectors $\langle 0, \ldots, 0 \rangle, \langle 1, \ldots, 1 \rangle$ respectively.

2.2 Economic Model

We adopt the terminology of Elliot and Golub [12] when possible. The salient pieces of our economy are defined as follows:

- The set of agents in the economy is given by $N = [n] = \{1, \ldots, n\}$. A nonempty subset of agents in the economy $C \subseteq N$ is called a *coalition*. The coalition $C = N$ is called the *grand coalition*.
- Each agent i chooses an *action* a_i, which can be any real number in the interval $[0, 1]$. An *outcome* or *point* is a vector $\mathbf{a} \in \mathbb{R}^n$ built by concatenating the actions of all agents.
- There is a continuous *utility function* $\mathbf{u} : [0, 1]^n \to [0, 1]^n$, which maps outcomes to a level of "utility" for each agent. In particular, agent i prefers outcome $\mathbf{a}$ to outcome $\mathbf{a}'$ iff $u_i(\mathbf{a}) > u_i(\mathbf{a}')$. The utility function has the following two properties:
  - *Positive Externalities*: whenever $\mathbf{a} \gneq \mathbf{a}'$ with $a_i = a'_i$, we have $u_i(\mathbf{a}) > u_i(\mathbf{a}')$. This assumption is what places us in the setting of public goods economies; intuitively, it states that an agent gains utility when other agents increase their production of public goods.
  - *Convex Preferences*: we assume that $\mathbf{u}$ is concave.[3] That is, for any outcomes $\mathbf{a}, \mathbf{a}'$ and any $\lambda \in [0, 1]$, we have $\mathbf{u}(\lambda\mathbf{a} + (1 - \lambda)\mathbf{a}') \geq \lambda\mathbf{u}(\mathbf{a}) + (1 - \lambda)\mathbf{u}(\mathbf{a}')$. This standard assumption corresponds to the economic principle of diminishing marginal returns.

2.3 Game Theory Definitions

We recap some well-known definitions from the game theory literature.

▶ **Definition 4** (Pareto Efficiency)**.** An outcome $\mathbf{a}$ is a *Pareto Improvement* on another outcome $\mathbf{a}'$ if $\mathbf{u}(\mathbf{a}) \gneq \mathbf{u}(\mathbf{a}')$. An outcome $\mathbf{a}$ is *Pareto Efficient* if there is no Pareto improvement on $\mathbf{a}$. The set of Pareto efficient outcomes is called the *Pareto Frontier*.

The main solution concept that will be discussed in this paper is *the core*:

▶ **Definition 5** (Deviation)**.** Given an outcome $\mathbf{a}$, an outcome $\mathbf{a}'$ is a *deviation* from $\mathbf{a}$ for a coalition C if $\mathbf{a}'_{N \setminus C} = \mathbf{0}$ and $\mathbf{u}_C(\mathbf{a}') > \mathbf{u}_C(\mathbf{a})$.

▶ **Definition 6** (The Core)**.** An outcome $\mathbf{a}$ is in the *core* of the economy if no coalition has a deviation from $\mathbf{a}$ (equivalently, $\mathbf{a}$ is *coalitionally stable*).

The next definition that will be useful in our proofs is the *projected economy*:

▶ **Definition 7.** Given an economy described by agents N and a utility function $\mathbf{u}$, the *projected economy* for a coalition C is the economy described by agents N and utility function $\mathbf{u}^C(\mathbf{a}_C)$, where

$$\mathbf{u}^C(\mathbf{a}_C) := \mathbf{u}_C(\mathbf{a}_C \cdot \mathbf{0}_{N \setminus C}).$$

In other words, the new $|C|$-dimensional utility function $\mathbf{u}^C$ is obtained by fixing the actions of $N \setminus C$ at 0, allowing any action for C, and then using the old utility function $\mathbf{u}$ to determine the utilities for C in the natural way. We suppress the superscript $\mathbf{u}^C$ when clear from context.

[3] Confusingly, when $\mathbf{u}$ is mathematically concave, one says that preferences are "economically convex" – hence, *convex preferences*.

▶ **Definition 8.** The *dominated set* of $\mathbf{a}$, denoted $\mathbf{D_a}$, is defined as:

$$\mathbf{D_a} := \{\mathbf{a}' \mid \mathbf{u}(\mathbf{a}) \geq \mathbf{u}(\mathbf{a}')\}.$$

In other words, $\mathbf{D_a}$ is the set of points that no agent prefers to $\mathbf{a}$. Note that this is an unusually weak definition of dominance, in the sense that (for example) $\mathbf{D_a}$ contains $\mathbf{a}$ itself.

3 A Topological Characterization of the Core

Our goal in this section is to prove the following structural theorem:

▶ **Theorem 9.** *Let* $\mathbf{a}$ *be an outcome in a public goods economy. Then* $\mathbf{a}$ *is in the core if and only if it is Pareto efficient, individually rational, and* $\mathbf{D_a}$ *is path connected.*

The vast majority of the technical depth of this theorem is tied up in the implication

$$\mathbf{a} \text{ is in the core } \longrightarrow \mathbf{D_a} \text{ is path connected.}$$

The remainder of this forwards implication ($\mathbf{a}$ is in the core $\rightarrow$ neither the grand coalition nor any singleton coalition has a deviation from $\mathbf{a}$) is extremely straightforward: $\mathbf{a}$ is individually rational iff each agent i prefers it to the outcome they can guarantee acting alone, which coincides with the notion that the singleton coalition $\{i\}$ has no deviation from $\mathbf{a}$. In our model, Pareto efficiency coincides with the notion that the grand coalition N has no deviation from $\mathbf{a}$:

▶ **Claim 10.** *Let* $\mathbf{a}$ *be an outcome. If there is an outcome* $\mathbf{a}'$ *satisfying* $\mathbf{a} \gneq \mathbf{a}'$ *(*$\mathbf{a} \lneq \mathbf{a}'$*) and* $\mathbf{u}(\mathbf{a}) \gneq \mathbf{u}(\mathbf{a}')$*, then there is an outcome* $\mathbf{a}''$ *satisfying* $\mathbf{a} > \mathbf{a}''$ *(*$\mathbf{a} < \mathbf{a}'$*) and* $\mathbf{u}(\mathbf{a}) > \mathbf{u}(\mathbf{a}'')$*.*

Proof. We will prove the claim for the case $\mathbf{a} \gneq \mathbf{a}'$; the case $\mathbf{a} \lneq \mathbf{a}'$ follows from a symmetric argument.

Choose an agent i for whom $u_i(a) > u_i(a')$, and then slightly increase a_i. Since $\mathbf{u}$ is continuous, if we increase a_i by a sufficiently small amount then we still have $u_i(a) > u_i(a')$. Additionally, by positive externalities we then have $\mathbf{u}(\mathbf{a}) > \mathbf{u}(\mathbf{a}')$. We can then slightly increase the actions of all agents, such that $\mathbf{a} > \mathbf{a}'$, but with sufficiently small increases we do not destroy the property that $\mathbf{u}(\mathbf{a}) > \mathbf{u}(\mathbf{a}')$. ◀

In this section, we will first give a complete proof of the (easier) backwards implication of Theorem 9, and then we sketch the proof of the forwards implication. Due to space constraints, a full proof of the forwards implication can be found in the full version of this paper.

3.1 Backwards Implication of Theorem 9

First:

▶ **Lemma 11.** *If* $\mathbf{a}$ *is Pareto efficient, then every deviation* $\mathbf{a}'$ *from* $\mathbf{a}$ *satisfies* $\mathbf{a}' \lneq \mathbf{a}$*.*

Proof. Let I be the set of agents i for which $a'_i > a_i$, and suppose towards a contradiction that I is nonempty.

Consider the point $\mathbf{a}''$ defined such that $\mathbf{a}''_{N\setminus I} := \mathbf{a}_{N\setminus I}$ and $\mathbf{a}''_I := \mathbf{a}'_I$. We then have $\mathbf{u}_{N\setminus I}(\mathbf{a}'') > \mathbf{u}_{N\setminus I}(\mathbf{a})$ by positive externalities, since these points differ only in that the (nonempty) coalition I has increased their actions. We also have $\mathbf{u}_I(\mathbf{a}'') \geq \mathbf{u}_I(\mathbf{a}') > \mathbf{u}_I(\mathbf{a})$, where the first inequality follows from positive externalities (since these points differ only in

that the coalition $N \setminus I$ has weakly increased their action), and the second follows from the fact that $\mathbf{a}'$ is a deviation from $\mathbf{a}$ for a coalition C with $I \subseteq C$ (since $\mathbf{a}_I > \mathbf{a}_I$).

We thus have $\mathbf{u}(\mathbf{a}'') > \mathbf{u}(\mathbf{a})$, which contradicts the fact that $\mathbf{a}$ is Pareto efficient. Thus I is empty and the lemma follows. ◀

Second:

▶ **Lemma 12.** *Suppose there is a path $P \subset \mathbb{R}^n_{\geq \mathbf{0}}$ with endpoints $\mathbf{x}, \mathbf{y}$ such that for any $\mathbf{p} \in P$ we have $\mathbf{u}(\mathbf{p}) \leq \mathbf{u}(\mathbf{a})$. If $\mathbf{a}'$ is a deviation from $\mathbf{a}$ for some coalition C satisfying $\mathbf{a}' \lneq \mathbf{x}$, then $\mathbf{a}'$ also must satisfy $\mathbf{a}' \lneq \mathbf{y}$.*

Proof. We walk along P from $\mathbf{x}$ towards $\mathbf{y}$ until we find the first point $\mathbf{p}$ with $p_i = a'_i \neq 0$ for some i. If we reach $\mathbf{y}$ before we find any such point $\mathbf{p}$, it follows that $a'_i = 0$ or $a'_i < x_i$ for all i, and so $\mathbf{a}' \lneq \mathbf{x}$, as claimed. Otherwise, we find such a point $\mathbf{p}$, and we argue towards a contradiction.

We have $\mathbf{p} \neq \mathbf{a}'$, since $\mathbf{u}_C(\mathbf{a}') \gneq \mathbf{u}_C(\mathbf{a})$ but $\mathbf{u}_C(\mathbf{p}) \leq \mathbf{u}_C(\mathbf{a})$. By construction we then have $\mathbf{p} \gneq \mathbf{a}'$. Since $p_i = a'_i$, by positive externalities we then have $u_i(p) > u_i(a')$. Since $a'_i \neq 0$ we have $i \in C$, and since $\mathbf{a}'$ is a deviation for C, this implies $u_i(a') > u_i(a)$. We then have $u_i(p) > u_i(a)$, which contradicts the assumption that $\mathbf{u}(\mathbf{p}) \leq \mathbf{u}(\mathbf{a})$. Therefore no such point $\mathbf{p}$ may be found. ◀

We can now show:

Proof of Theorem 9, Backwards Implication. Assume that $\mathbf{a}$ is robust to deviations by the grand coalition or any singleton coalition, and that $\mathbf{D_a}$ is path connected. Our goal is now to show that $\mathbf{a}$ is in the core.

By Claim 10, the property that the grand coalition has no deviation from $\mathbf{a}$ implies that $\mathbf{a}$ is Pareto efficient. Thus, by Lemma 11 any deviation $\mathbf{a}'$ from $\mathbf{a}$ satisfies $\mathbf{a}' \lneq \mathbf{a}$. Since no singleton coalition has a deviation from $\mathbf{a}$ we have $\mathbf{0} \in \mathbf{D_a}$, and since $\mathbf{D_a}$ is path connected there is a path contained in $\mathbf{D_a}$ with endpoints $\mathbf{a}, \mathbf{a}$. Thus, by Lemma 12, we further have that a deviation $\mathbf{a}'$ must satisfy $\mathbf{a}' \lneq \mathbf{0}$. Since no such point exists, it follows that no deviations from $\mathbf{a}$ exist, and so $\mathbf{a}$ is a core outcome. ◀

3.2 Sketch of Forwards Implication of Theorem 9

We will denote by $\mathbf{d_v u(a)}$ the one-sided directional derivative of $\mathbf{u}$ at $\mathbf{a}$ in the direction $\mathbf{v}$. In other words:

$$\mathbf{d_v u(a)} := \lim_{\lambda \to 0^+} \frac{\mathbf{u}(\mathbf{a} + \lambda \mathbf{v}) - \mathbf{u}(\mathbf{a})}{\lambda}.$$

A nontrivial but standard fact from analysis is that, since $\mathbf{u}$ is concave and well-defined everywhere, this limit is well-defined for all $\mathbf{a}$, except when excluded by a boundary condition (e.g. if $v_i < 0$ but $a_i = 0$ for some agent i) – see [21].

Our key lemma is:

▶ **Lemma 13.** *At any outcome $\mathbf{0} < \mathbf{a} < \mathbf{1}$, exactly one of the following three conditions holds:*

1. *There exist directions $\mathbf{v}_{up} > \mathbf{0}, \mathbf{v}_{down} < \mathbf{0}$ such that $\mathbf{d}_{\mathbf{v}_{up}} \mathbf{u}(\mathbf{a}) > \mathbf{0}$ and $\mathbf{d}_{\mathbf{v}_{down}} \mathbf{u}(\mathbf{a}) < \mathbf{0}$,*
2. *There exist directions $\mathbf{v}_{up} > \mathbf{0}, \mathbf{v}_{down} < \mathbf{0}$ such that $\mathbf{d}_{\mathbf{v}_{up}} \mathbf{u}(\mathbf{a}) \leq \mathbf{0}$ and $\mathbf{d}_{\mathbf{v}_{down}} \mathbf{u}(\mathbf{a}) \leq \mathbf{0}$, or*
3. *There exist directions $\mathbf{v}_{up} > \mathbf{0}, \mathbf{v}_{down} < \mathbf{0}$ such that $\mathbf{d}_{\mathbf{v}_{up}} \mathbf{u}(\mathbf{a}) < \mathbf{0}$ and $\mathbf{d}_{\mathbf{v}_{down}} \mathbf{u}(\mathbf{a}) > \mathbf{0}$.*

The three categories of Lemma 13 carry a useful geometric intuition. Specifically:

▶ **Lemma 14.** *The points in the second category of Lemma 13 are precisely the Pareto Frontier.*

The proofs of these two lemmas are quite technical, and can be found in the full version of this paper. With these in mind, we define

▶ **Definition 15.** We will say that a point in the first category of Lemma 13 is *below the Pareto Frontier*, and a point in the third category of Lemma 13 is *above the Pareto Frontier* (and by Lemma 14, the second category of points in Lemma 13 are on the Pareto Frontier).

The geometric intuition behind this definition is that, starting from a point $\mathbf{x}$ in the first category, one can continuously follow the gradient $\mathbf{v}_{up}$ to eventually obtain a Pareto efficient Pareto improvement $\mathbf{x}' > \mathbf{x}$ (we do not prove this fact formally; it is perhaps useful intuition but not essential to our main results). Similarly, starting from $\mathbf{x}$ in the third category, we can continuously follow the gradient $\mathbf{v}_{down}$ to obtain a Pareto efficient Pareto improvement $\mathbf{x}' < \mathbf{x}$.

We then show:

Proof Sketch of Theorem 9, Forwards Implication (Proof in full version). Suppose $\mathbf{a}$ is a core outcome, and our goal is to show path connectedness of $\mathbf{D_a}$. First, we note that $\mathbf{0} \in \mathbf{D_a}$, since otherwise a singleton coalition can deviate from $\mathbf{a}$ (and $\mathbf{a}$ is in the core, so no such deviation is possible). To show path connectedness of $\mathbf{D_a}$, we consider an arbitrary point $\mathbf{x} \in \mathbf{D_a}$ and construct a path in $\mathbf{D_a}$ from $\mathbf{x}$ to $\mathbf{0}$, thus implying that any two such points $\mathbf{x}, \mathbf{x}' \in \mathbf{D_a}$ have a connecting path in $\mathbf{D_a}$ via $\mathbf{0}$.

We show the existence of the $\mathbf{x} \rightsquigarrow \mathbf{0}$ path with a careful repeated application of Lemma 13. Informally speaking, we progressively slide $\mathbf{x} > \mathbf{0}$ a little bit closer to $\mathbf{0}$ while maintaining the property $\mathbf{x} \in \mathbf{D_a}$. If we ever hit $x_i = 0$ for some agent i, then we restrict our attention to the projected economy discluding agent i and continue. If we eventually exclude all agents in this manner, then we have $\mathbf{x} = \mathbf{0}$ and the process is complete. Otherwise, suppose towards a contradiction that at some $\mathbf{x}$, we cannot slide $\mathbf{x}$ any closer to $\mathbf{0}$ while maintaining $\mathbf{x} \in \mathbf{D_a}$. We make two observations here: (1) $\mathbf{x}$ must be above the Pareto frontier (else we could slide $\mathbf{x}$ in the appropriate direction $\mathbf{v}_{down}$) and so it belongs to the third; and (2) for all agents i still being considered, we have $u_i(x) = u_i(a)$ (else, by the positive externalities assumption, we can unilaterally decrease the action of agent i without destroying $\mathbf{x} \in \mathbf{D_a}$). Hence, by moving $\mathbf{x}$ slightly in the direction $\mathbf{v}_{down}$ (which *improves* the utility of all agents being considered), we have $u_i(x) > u_i(a)$ for all agents being considered, and so the new $\mathbf{x}$ is a deviation from $\mathbf{a}$. Since we have assumed that $\mathbf{a}$ is a core outcome, this is a contradiction, and so the process of sliding $\mathbf{x}$ towards $\mathbf{0}$ can never get stuck in this way. ◀

3.3 Connection to Lindahl Equilibria

Before proceeding towards our algorithm, we take a brief detour in this subsection to observe an interesting implication of Theorem 9 that helps illustrate its broader appeal. Elliot and Golub [12] show the following result:

▶ **Theorem 16** ([12]). *The Lindahl equilibria of a public goods economy with a differentiable utility function are precisely the outcomes* $\mathbf{a}$ *for which* $\mathbf{d_a u(a)} = \mathbf{0}$.

They phrase this theorem in different language related to the "benefits network" of the economy, but this formulation will suit our purposes better. We refer the reader to their paper for a more in-depth discussion of the economic role of the Lindahl equilibria.

Combining Theorem 16 with our machinery for Theorem 9, we obtain:

▶ **Theorem 17.** *In a public goods economy with a differentiable utility function, the Lindahl equilibria are precisely the core outcomes* $\mathbf{a}$ *whose membership can be certified by examining only local information at* $\mathbf{a}$.

The proof of this theorem will use Theorem 16 as a black box, and so we will not actually need to appeal to the formal definition of the Lindahl equilibria in its proof.

Proof. If $\mathbf{a}$ is a Lindahl equilibrium, then by Theorem 16 we have $\mathbf{d_a}\,\mathbf{u}(\mathbf{a}) = \mathbf{d_{-a}}\,\mathbf{u}(\mathbf{a}) = \mathbf{0}$ (the first equality comes from the assumption that $\mathbf{u}$ is differentiable). We claim that any $\mathbf{a}$ satisfying $\mathbf{d_a}\,\mathbf{u}(\mathbf{a}) = \mathbf{d_{-a}}\,\mathbf{u}(\mathbf{a}) = \mathbf{0}$ is in the core, and thus its core membership can be verified by examining only these local derivatives. First, note that $\mathbf{a}$ is Pareto efficient by Lemma 14. Therefore, by Lemma 11, any possible deviation $\mathbf{a}'$ from $\mathbf{a}$ satisfies $\mathbf{a}' \lneq \mathbf{a}$. Now let P be the line segment from $\mathbf{0}$ to $\mathbf{a}$. By the assumption of concavity and the fact that $\mathbf{d_{-a}}\,\mathbf{u}(\mathbf{a}) = \mathbf{0}$, we have $\mathbf{u}(\mathbf{p}) \leq \mathbf{u}(\mathbf{a})$ for all $\mathbf{p} \in P$. Thus, by Lemma 12, we have $\mathbf{a}' \lneq \mathbf{0}$ and so $\mathbf{a}'$ cannot exist and $\mathbf{a}$ is a core outcome.

Now suppose that $\mathbf{a}$ is not a Lindahl equilibrium, and so $\mathbf{d_{-a}}\,\mathbf{u}(\mathbf{a}) \neq \mathbf{0}$. If we have $\mathbf{d_{-a}}\,\mathbf{u}(\mathbf{a}) \lneq \mathbf{0}$, then we have $\mathbf{d_a}\,\mathbf{u}(\mathbf{a}) \gneq \mathbf{0}$ (by differentiability) which implies that $\mathbf{a}$ is not Pareto efficient, and hence is not a core outcome. On the other hand, suppose we have $\mathbf{d_{-a}}\,\mathbf{u}(\mathbf{a}) \gneq \mathbf{0}$. In this case, it is impossible to distinguish $\mathbf{u}$ from the utility function $\mathbf{u}'$ that is affine-linear everywhere and agrees with $\mathbf{u}$ at $\mathbf{a}$ using solely local information. Note that $\mathbf{a}$ is *not* in the core of the economy defined by $\mathbf{u}'$, since we have $u_i(\mathbf{0}) > u_i(a)$ for whichever agent i satisfies $d_{-a}\,u_i(\mathbf{a}) > 0$. Thus, if $\mathbf{a}$ is in fact in the core of the economy defined by $\mathbf{u}$, we will need to inspect non-local information about the economy to differentiate $\mathbf{u}$ from $\mathbf{u}'$. ◀

We note that it is possible to prove Theorem 17 as a corollary directly from Theorem 9, but this proof using the underlying machinery is simpler.

4 Algorithm for Testing Core Membership

Our main algorithmic result is:

▶ **Theorem 18.** *Given an outcome* $\mathbf{a}$ *in a public goods economy, there is an algorithm (in the real-RAM model) that decides whether or not* $\mathbf{a}$ *is in the core by solving* $O(n)$ *convex optimization problems and using* $O(n)$ *additional computation time.*

The algorithm is fairly straightforward. We maintain an "active coalition" C_A throughout, as well as a proof that any agent $i \notin C_A$ must play action $a'_i = 0$ in any deviation $\mathbf{a}'$ from $\mathbf{a}$. It is thus safe to assume that any deviating coalition C satisfies $C \subseteq C_A$. Initially $C_A \leftarrow N$, so this invariant is trivially satisfied. After each round, we either find a deviation for C_A from $\mathbf{a}$, or we remove one new agent from C_A. Thus, if we make it n rounds without finding a deviation, then we have $C_A = \emptyset$ and so no deviation from $\mathbf{a}$ is possible.

4.1 Preprocessing: Confirm Pareto Efficiency of a

Before starting the main algorithm, we run the following two programs, with the purpose of testing whether or not $\mathbf{a}$ is Pareto efficient.

▶ **Program 1.** Choose $\mathbf{v}$ to maximize $\min_i d_v\,u_i(a)$
Subj. to $\mathbf{v} \geq \mathbf{0}, \sum_i v_i = 1$

▶ **Program 2.** Choose $\mathbf{v}$ to maximize $\min_i d_v\, u_i(a)$
Subj. to $\mathbf{v} \leq \mathbf{0}, \sum_i v_i = 1$

Note that the concavity of the optimized function $f(v) := \min_i d_v\, u_i(a)$ is immediate from the concavity of $\mathbf{u}$. By Lemma 13, we may immediately conclude that $\mathbf{a}$ is not Pareto efficient (and thus not in the core) iff either of these programs optimizes at a point $\mathbf{v}^*$ satisfying $f(v^*) > 0$. Otherwise, we proceed with the knowledge that $\mathbf{a}$ is Pareto efficient. A key advantage of this is that, by Lemma 11, we may now restrict our search for a deviation $\mathbf{a}'$ to the bounded box $\mathbf{0} \leq \mathbf{a}' \leq \mathbf{a}$. This opens up the ability to use convex programming algorithms, which typically require bounded domains, in the remainder of the algorithm.[4]

4.2 Main Loop: Shrinking C_A

Each of the n rounds of the algorithm consists of three steps. First, we restrict our attention to the projected economy for the coalition C_A. Second, we run the following program:

▶ **Program 3.** Choose $\mathbf{x}$ to maximize $\min_i u_i(x) - u_i(a)$
Subj. to $\mathbf{0} \leq \mathbf{x} \leq \mathbf{a}_{C_A}$

Let $\mathbf{x}^*$ be a maximizing point of Program 3. We have:

▶ **Lemma 19.** *Either* $\mathbf{u}^{C_A}(\mathbf{x}^*) > \mathbf{u}_{C_A}(\mathbf{a})$ *or* $\mathbf{u}^{C_A}(\mathbf{x}^*) \leq \mathbf{u}_{C_A}(\mathbf{a})$*, and* $\mathbf{x}^*$ *is Pareto efficient.*[5]

Proof. First we argue Pareto efficiency. If $\mathbf{x}'$ is a Pareto improvement on $\mathbf{x}^*$, then by Claim 10 there is another point $\mathbf{x}''$ with $\mathbf{u}(\mathbf{x}'') > \mathbf{u}(x^*)$. This $\mathbf{x}''$ would be a superior maximizing point for Program 3, so there can be no Pareto improvement on $\mathbf{x}^*$.

Next, let $i := \arg\max_i u_i(x^*) - u_i(a)$ and $j := \arg\min_j u_j(x^*) - u_j(a)$. If $u_i(x^*) - u_i(a) > u_j(x^*) - u_j(a)$, then (by the same argument used in Claim 10) we can again obtain a superior maximizing point $\mathbf{x}^{**}$ by slightly increasing the action of agent i from $\mathbf{x}^*$. Thus we have $u_i(x^*) - u_i(a) = u_j(x^*) - u_j(a)$, and it follows that either $\mathbf{u}(\mathbf{x}^*) > \mathbf{u}(\mathbf{a})$ or $\mathbf{u}(\mathbf{x}^*) \leq \mathbf{u}(\mathbf{a})$. ◀

In the former case where $\mathbf{u}(\mathbf{x}^*) > \mathbf{u}(\mathbf{a})$, it follows that $\mathbf{x}^*$ is a deviation from $\mathbf{a}$ for the coalition C_A, so we may halt the algorithm. Otherwise, we have $\mathbf{u}(\mathbf{x}^*) \leq \mathbf{u}(\mathbf{a})$. We then observe:

▶ **Lemma 20.** *If* $\mathbf{u}^{C_A}(\mathbf{x}^*) \leq \mathbf{u}_{C_A}(\mathbf{a})$*, then in the full (non-projected) economy, any deviation* $\mathbf{a}'$ *from* $\mathbf{a}$ *for a coalition* $C \subseteq C_A$ *satisfies* $\mathbf{a}'_{C_A} \lneq \mathbf{x}^*$.

Proof. The deviation $\mathbf{a}'$ satisfies $\mathbf{a}'_{N \setminus C} = \mathbf{0}$ and $\mathbf{u}_C(\mathbf{a}') > \mathbf{u}_C(\mathbf{a}) \geq \mathbf{u}_C(\mathbf{x}^*)$. It follows that $\mathbf{a}'_C$ is also a deviation for C from $\mathbf{x}^*$ in the projected economy for C_A. The claim is then immediate from Lemma 11. ◀

One step remains. We run:

▶ **Program 4.** Choose $\mathbf{v}$ to maximize $\min_i d_v\, u_i(x^*)$
Subj. to $\mathbf{v} \leq \mathbf{0}, \sum_i v_i = 1$

[4] This detail is precisely why we use Programs 1 and 2 to check the Pareto efficiency of $\mathbf{a}$, rather than the ostensibly simpler method of searching for $\mathbf{x}^*$ that maximizes $\min_i u_i(x^*) - u_i(a)$: the latter method requires a search for $\mathbf{x}^*$ over an unbounded search space, which rules out many popular methods of convex optimization that we wish to keep available.

[5] Note that these statements hold specifically in the projected economy for C_A.

Algorithm 1: Testing Core Membership of $\mathbf{a}$.

Let $\mathbf{v}_1^* \leftarrow$ output of Program 1;
Let $\mathbf{v}_2^* \leftarrow$ output of Program 2;
if $\mathbf{d}_{\mathbf{v}_1^*}\mathbf{u}(\mathbf{a}) > \mathbf{0}$ *or* $\mathbf{d}_{\mathbf{v}_1^*}\mathbf{u}(\mathbf{a}) > \mathbf{0}$ **then**
  return "$\mathbf{a}$ *is not in the core*";
end
$C_A \leftarrow N$;
while $C_A \neq \emptyset$ **do**
  $\mathbf{x}^* \leftarrow$ output of Program 3;
  if $\mathbf{u}^{C_A}(\mathbf{x}^*) > \mathbf{u}_{C_A}(\mathbf{a})$ **then**
    return "$\mathbf{a}$ *is not in the core*";
  end
  $\mathbf{v}^* \leftarrow$ output of Program 4;
  $C_A \leftarrow C_A \setminus \{\arg\min_{i \in C_A} x_i^*/v_i^*\}$;
end
return "$\mathbf{a}$ *is in the core*";

Let $\mathbf{v}^*$ be a point that maximizes Program 4. We have

▶ **Lemma 21.**

$$\mathbf{d}_{\mathbf{v}^*}\mathbf{u}(\mathbf{x}^*) \leq \mathbf{0}\,.$$

The proof is very similar to the proof of Lemma 19, so we omit it for now. We then finally have:

▶ **Lemma 22.** *Let* $i := \arg\min_i x_i^*/v_i^*$. *Then any deviation* $\mathbf{a}'$ *from* $\mathbf{a}$ *has* $a_i' = 0$.

Proof. By Lemma 20, we have $\mathbf{a}'_{C_A} \lneq \mathbf{x}^*$. Let P be the line segment starting at $\mathbf{x}^*$, extending in the direction $\mathbf{v}^*$ until a point $\mathbf{p}^*$ is reached where $p_i^* = 0$ for some agent i; note that this will specifically be $i = \arg\min_i x_i^*/v_i^*$. By concavity, all $\mathbf{p} \in P$ satisfies $\mathbf{u}(\mathbf{p}) \leq \mathbf{u}(\mathbf{x}^*) \leq \mathbf{u}(\mathbf{a})$. Noting once again that $\mathbf{a}'_{C_A}$ is a deviation for C from $\mathbf{x}^*$ in the projected economy for C_A, it follows from Lemma 12 that $\mathbf{a}'_{C_A} \lneq \mathbf{p}^*$, and so $a_i' = 0$. ◀

With this in mind, the final step in the loop is to delete i from C_A and repeat. After n repetitions, we have $C_A = \emptyset$, so we may halt the algorithm and report that $\mathbf{a}$ is in the core.

4.3 Algorithm Pseudocode

To recap the algorithm, which has been interspersed with proofs of correctness above, we give full pseudocode here.

4.4 Conclusion

Our algorithm implies that core membership testing is efficient under any utility function that admits quick solving of convex programs as described above. However, it may still be desirable to test core membership as best as possible when the underlying utility function is either unknown or badly behaved and so exact convex optimization is impossible. Our algorithm can indeed be adapted to this effect, with a few significant points of caution, by substituting in modern approximate optimization algorithms. Due to space constraints, we defer a discussion of this point to the full version of the paper.

References

1 N. Allouch. The cost of segregation in social networks. *Queen Mary Working Paper*, 2013.
2 N. Allouch. On the private provision of public goods on networks. *Journal of Economic Theory*, pages 527–552, 2015.
3 C. Ballester, A. Calvó-Armengol, and Y. Zenou. Who's who in networks. wanted: The key player. *Econometrica*, 74:1403–1417, 2006.
4 Y. Bramoulle and R. Kranton. Public goods in networks. *Journal of Economic Theory*, 135:478–494, 2007.
5 Parkash Chander and Henry Tulkens. The core of an economy with multilateral environmental externalities. *International Journal of Game Theory*, 26:379–401, 1997.
6 V. Conitzer and T. Sandholm. Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proc. National Conference on Artificial Intelligence (AAAI)*, pages 219–225, 2004.
7 V. Conitzer and T. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, pages 607—619, 2006.
8 Imma Curriel. *Cooperative Game Theory and Applications*. Springer US, 1 edition, 1997.
9 X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19:257, 1994.
10 F. Y. Edgeworth. Mathematical psychics: An essay on the mathematics to the moral sciences. *Reprinted in Diamond, M.A.*, 1881.
11 E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. *Math. Log. Q.*, 2009.
12 Matthew Elliot and Benjamin Golub. A network approach to public goods. In *Proc. 14th Electronic Commerce (EC)*, pages 377–378, 2013.
13 U. Faigle, S. Fekete, W. Hochstattler, and W. Kern. On the complexity of testing membership in the core of min-cost spanning trees. *Internat. J. Game Theory*, 26:361—366, 1997.
14 D. K. Foley. Lindahl's solution and the core of an economy with public goods. *Econometrica*, 38:66–72, 1970.
15 D. B. Gillies. Solutions to general non-zero-sum games. *Contributions to the Theory of Games IV*, pages 47–85, 1959.
16 M. Goemans and M. Skutella. Cooperative facility location games. In *Symposium on Discrete Algorithms (SODA)*, pages 76–85, 2000.
17 Benjamin Golub. Personal correspondence, 2012.
18 G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of the core over coalition structures. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.
19 S. Ieong and Y. Shoham. Marginal contribution nets: A compact representation scheme for coalitional games. In *Electronic Commerce (EC)*, pages 193–202, 2005.
20 S. Kintali, L. Poplawski, R. Rajaraman, R. Sundaram, and S. Teng. Reducibility among fractional stability problems. In *Proc. 50th FOCS*, pages 283–292, 2009.
21 Jonathan Lewin. *An interactive introduction to mathematical analysis*. Cambridge University Press, 2003.
22 Y. Li and V. Conitzer. Complexity of stability-based solution concepts in multi-issue and mc-net cooperative games. In *Proc. 2014 international conference on Autonomous agents and multi-agent systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, 2014.
23 Bezalel Peleg and Peter Sudhölter. *Introduction to the Theory of Cooperative Games*. Springer, 2 edition, 10 2007.

24 Paul Samuelson. The pure theory of public expenditure. *Review of Economics and Statistics*, 37(4):350–356, 1954.

25 H. Scarf. The core of an n person game. *Econometrica*, 1967.

26 S. Sung and D. Dimitrov. On core membership testing for hedonic coalition formation games. *Operations Research Letters*, 35:155–158, 2007.

Revenue Maximization in Stackelberg Pricing Games: Beyond the Combinatorial Setting

Toni Böhnlein[1], Stefan Kratsch[2], and Oliver Schaudt[3]

1 **Universität zu Köln, Institut für Informatik, Cologne, Germany**
`boehnlein@zpr.uni-koeln.de`
2 **Universität Bonn, Institut für Informatik, Bonn, Germany**
`kratsch@cs.uni-bonn.de`
3 **Universität zu Köln, Institut für Informatik, Cologne, Germany**
`schaudto@uni-koeln.de`

Abstract

In a Stackelberg Pricing Game a distinguished player, the *leader*, chooses prices for a set of items, and the other players, the *followers*, each seeks to buy a minimum cost feasible subset of the items. The goal of the leader is to maximize her revenue, which is determined by the sold items and their prices. Most previously studied cases of such games can be captured by a combinatorial model where we have a base set of items, some with fixed prices, some priceable, and constraints on the subsets that are feasible for each follower. In this combinatorial setting, Briest et al. and Balcan et al. independently showed that the maximum revenue can be approximated to a factor of $H_k \sim \log k$, where k is the number of priceable items.

Our results are twofold. First, we strongly generalize the model by letting the follower minimize any continuous function plus a linear term over any compact subset of $\mathbb{R}^n_{\geq 0}$; the coefficients (or *prices*) in the linear term are chosen by the leader and determine her revenue. In particular, this includes the fundamental case of linear programs. We give a tight lower bound on the revenue of the leader, generalizing the results of Briest et al. and Balcan et al. Besides, we prove that it is strongly NP-hard to decide whether the optimum revenue exceeds the lower bound by an arbitrarily small factor. Second, we study the parameterized complexity of computing the optimal revenue with respect to the number k of priceable items. In the combinatorial setting, given an efficient algorithm for optimal follower solutions, the maximum revenue can be found by enumerating the 2^k subsets of priceable items and computing optimal prices via a result of Briest et al., giving time $O(2^k|I|^c)$ where $|I|$ is the input size. Our main result here is a W[1]-hardness proof for the case where the followers minimize a linear program, ruling out running time $f(k)|I|^c$ unless $\mathsf{FPT} = \mathsf{W}[1]$ and ruling out time $|I|^{o(k)}$ under the Exponential-Time Hypothesis.

1998 ACM Subject Classification G.2.0 [Discrete Mathematics] General

Keywords and phrases Algorithmic pricing, Stackelberg games, Approximation algorithms, Revenue maximization, Parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.46

1 Introduction

Pricing problems are fundamental in both economics and mathematical optimization. In this paper we study such pricing problems formulated as games, which are usually called *Stackelberg Pricing Games* [18]. In our setting, in order to maximize her revenue one player chooses prices for a number of items and one or several other players are interested in buying these items. Following the standard terminology, the player to choose the prices is called the *leader* while the other players are called *followers*. Depending on the follower's preferences,

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 46; pp. 46:1–46:13

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

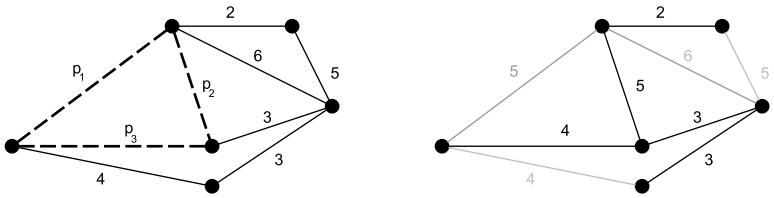


Figure 1 An instance of the Stackelberg Minimum Spanning Tree Game.

computing optimal prices can be a computational non-trivial problem. In a setting where followers have valuations over individual items only, the problem is simple. If, however, valuations become more complex, e.g., over whole subsets of items, pricing problems become much harder–also in a formal sense.

Largely the literature has focused on what we call the *combinatorial setting*: there is a set Y of items and one follower seeks to buy a feasible subset. Some of the items have fixed costs, the others have prices that are chosen by the leader. If the follower buys a feasible subset $S \subseteq Y$ of the items, he has to pay the sum of the fixed costs of the elements of S, plus the leader's prices of the bought elements. The leader's revenue is the sum of the prices of the priceable items in S. This can also be captured by defining a solution space X containing 0/1-vectors corresponding to the feasible subsets S of Y. The goal of the follower is then to minimize a given additive function $f\colon X \to \mathbb{R}$ that depends on both fixed and leader-chosen prices.

So-called Stackelberg Network Pricing Games became popular when Labbé et al. [15] used them to model road toll setting problems. In this game, the leader chooses prices for a subset of *priceable* edges in a network graph while the remaining edges have fixed costs. Each follower has a pair of vertices (s, t) and wants to buy a minimum cost path from s to t, taking into account both the fixed costs and the prices chosen by the leader. The work of Labbé et al. led to a series of studies of the Stackelberg Shortest Path Game. Roche et al. [16] showed that the problem is NP-hard, even if there is only one follower, and it has later been shown to be APX-hard [5, 14]. More recently, other combinatorial optimization problems were studied in their Stackelberg pricing version. For example, Cardinal et al. [9, 10] studied the Stackelberg Minimum Spanning Tree Game, proving APX-hardness and giving some approximation results. Moreover, a special case of the Stackelberg Vertex Cover Game in bipartite graphs has been shown to be polynomially solvable by Briest et al. [7].

To get more familiar with the setting, we briefly discuss an example of the Stackelberg Minimum Spanning Tree Game.

▶ **Example 1.** The left hand side of Figure 1 depicts an instance of the problem. Here the leader can choose the prices p_1, p_2 and p_3 for the dashed edges, while the solid edges have fixed costs as displayed. To motivate the problem, think of the vertices as hubs in a network and of the edges as data connections. In this scenario, the followers are Internet Service Providers and want to connect all the hubs at minimum cost, thus want to compute a minimum spanning tree. The leader owns the dashed connections and wants to set prices, that yield a large revenue. Furthermore, there are competitors who own the solid connections and it is known how much they charge for their usage.

On the right hand side, an optimal pricing ($p_1 = p_2 = 5$ and $p_3 = 4$) and a corresponding minimum spanning tree are depicted. Thus the leader's revenue amounts to 9, which can be verified to be maximum. Observe that we can compute a minimum spanning tree without

any priceable edges; otherwise the leader's revenue is unbounded. In this example, the total cost of such a minimum spanning tree is 17. In contrast, if we set all prices to 0 and let the follower compute a minimum spanning tree, it has a total cost of 8. The difference of these two values, $17 - 8 = 9$, is an upper bound on the revenue of the leader, as explained later. This upper bound, which we denote by R, is sometimes called the optimal *social welfare* and will be important for our approximation result.

An important contribution to the study of Stackelberg Games was the discovery by Briest et al. [7]. They show that the optimal revenue can be approximated surprisingly well using a single-price strategy. For a single-price strategy the leader sets the same price for all of her priceable items. Basically, their result says the following: In any Network Pricing Game with k priceable items, there is some $\lambda \in \mathbb{R}_{\geq 0}$ such that, when assigning the price of λ to all priceable items at once, the obtained revenue is only a factor of H_k away from the optimal revenue. Here, $H_k = \sum_{i=1}^{k} 1/i$ denotes the *k-th harmonic number*. This discovery has been made independently, in a slightly different model, by Balcan et al. [2]. Actually, in both papers [2, 7] a stronger fact is proven: The single-price strategy yields a revenue that is at least R/H_k, where R is a natural upper bound on the optimal revenue. The definition of R was sketched in the example above, and is formally laid out later.

Our results. Our work focuses on pushing the knowledge on Stackelberg Pricing Games beyond the well-studied combinatorial setting, in order to capture more complex problems of the leader. This is motivated by the simple fact that the combinatorial setting is too limited to even model, e.g., a follower that has a minimum cost flow problem—a crucial problem in both, combinatorial optimization and algorithmic game theory. More generally, we might want to be able to give bounds and algorithms in the case when the follower has an arbitrary linear or even convex program. For example, the follower might have a production problem in which he needs to buy certain materials from the leader, but such pricing problems haven't been discussed in the literature so far.

We prove an approximation result that applies even to a setting generalizing linear and convex programs. In our model, the follower minimizes a continuous function f over a compact set of feasible solutions $x \in X \subseteq \mathbb{R}^n_{\geq 0}$. For some of the variables, say x_1 up to x_k, the leader can choose a price vector $p \in \mathbb{R}^k$. Now the follower chooses a vector $x \in X$ that minimizes his objective function $f(x) + \sum_{i=1}^{k} p_i x_i$. We remark that if X is a set containing 0/1-vectors only, then we are back to the classical combinatorial setting. The result of Briest et al. can be transferred to the case when f is non-additive, in view of their original proof. Moreover if X is a polytope and f is additive, the follower minimizes a linear program, which is an important special case.

In Section 2, we formally introduce this more general model and prove the following results.

- **(i)** The maximum revenue obtainable by the leader can be approximated to a logarithmic factor using a single-price strategy. This generalizes the above mentioned result of Briest et al. [7] not only to linear programs but to any kind of follower that is captured by our model.
- **(ii)** The analysis of point (i) is tight. There is a family of instances for which the single-price strategy yields maximum revenue. And this revenue meets the bound of point (i).
- **(iii)** It is strongly NP-hard to decide whether one can achieve a revenue that is only slightly larger than the one guaranteed by the single-price strategy. This holds true even in a very restricted combinatorial setting.

The second part of the paper deals with the parameterized complexity of Stackelberg Pricing Games (Section 3). To the best of our knowledge, the only result in this direction is

an XP-algorithm by Cardinal et al. [10] for the Stackelberg Minimum Spanning Tree Game in graphs of bounded treewidth.

In contrast to structural parameters like the treewidth of the input graph, we consider the complexity of the pricing problem when parameterized by the number of priceable variables (or items in the combinatorial setting). Our main result in this part is a W[1]-hardness proof for the case that the optimization problem of the follower is a linear program, which is arguably one of the most interesting cases that does not fit into the combinatorial setting. This rules out algorithms of running time $f(k)|I|^c$ unless FPT = W[1] for any function f and polynomial $|I|^c$ of the input size; it also rules out running time $|I|^{o(k)}$ under the Exponential-Time Hypothesis of Impagliazzo et al. [13]. This intractability result is complemented by a fairly simple FPT-algorithm with running time $O(2^k|I|^c)$ for any Stackelberg Game that fits into the combinatorial model, when provided with an efficient algorithm for finding optimal follower solutions. The algorithm enumerates all subsets of priceable items and applies a separation argument of Briest et al. [7] to compute optimal leader prices and revenue.

Related work. Most important for our work are the approximation results due to Briest et al. [7] and Balcan et al. [2], which were discussed above.

A larger body of work focuses on specific network problems in their Stackelberg Game version. Briest et al. [7] give a polynomial time algorithm for a special case of the Stackelberg Bipartite Vertex Cover Game. An algorithm with improved running time was later given by Baïou and Barahona [1]. As mentioned, Labbé et al. [15] use the Stackelberg Shortest Path Game to model road toll setting problems. They establish NP-hardness and use LP bilevel formulations to solve small instances. A combinatorial approximation algorithm with the same logarithmic approximation guarantee as the single-price strategy was given by Roch et al. [16]. Moreover, a lower bound on the approximability is due to Briest et al. [5]: they show that the Stackelberg Shortest Path Game is NP-hard to approximate within a factor of less than 2. This is an improvement over previous results by Joret [14] showing APX-hardness. Further research on the Stackelberg Shortest Path Game can be found in a survey by van Hoesel [17]. A similar problem, the Stackelberg Shortest Path Tree Game, is studied by Bilo et al. [4]. They give an NP-hardness proof and develop an efficient algorithm assuming that the number of priceable edges is constant. Later their algorithm was improved by Cabello [8].

Cardinal et al. [9] proved several positive approximation results for the Stackelberg Minimum Spanning Tree Game. In the same paper, they proved that the revenue maximization for this game is APX-hard and strongly NP-hard. We make use of their reduction in the proof of Theorem 7. Furthermore, Cardinal et al. [10] prove that this game remains NP-hard if the instances are planar graphs. However, the problem becomes polynomial-time solvable on graphs of bounded treewidth. Bilo et al. [3] consider the Stackelberg Minimum Spanning Tree Game for complete graphs.

Briest et al. [6] consider Stackelberg Games where the follower's optimization problem cannot be solved to optimality. Instead the follower uses a known approximation algorithm. They show that the Stackelberg Knapsack Game is NP-hard if the follower uses a greedy 2-approximate algorithm, and derive a $2 + \epsilon$ approximation algorithm. Furthermore, the revenue maximization problem can be solved efficiently in the Stackelberg Vertex Cover Game if the follower implements a primal-dual approximation.

2 Approximability of Stackelberg Pricing Games

In this section we first introduce the model in its full generality. Then we give a tight approximation result on the maximum revenue using a single-price strategy. We complement

this with a hardness proof by showing that deciding whether one can achieve a revenue that is only slightly larger than the one guaranteed by the single-price strategy is strongly NP-hard.

Our model. Let k and n be some natural numbers with $k \leq n$. The optimization problem of the follower is the following: He minimizes a continuous function $f : \mathbb{R}^n \to \mathbb{R}$ over his set of feasible solutions $X \subseteq \mathbb{R}^k_{\geq 0} \times \mathbb{R}^{n-k}_{\geq 0}$. The only restriction we put on X is that we require it to be a compact set, i.e., bounded and closed under limits.

The first move of the Stackelberg Pricing Game is made by the *leader*: She chooses a *price vector* $p \in \mathbb{R}^k$. Now the second move is made by the *follower*: He chooses an optimal solution (x^*, y^*) of the program

$$\begin{aligned} \min \quad & p^T x + f(x, y) \\ \text{s.t.} \quad & (x, y) \in X. \end{aligned}$$

The *revenue* of the leader is then given by the value $p^T x^*$. This value is her objective function and it is to be maximized. We remark that this problem has a bilevel structure.

To avoid technicalities, we make the following optimistic assumption: If the follower has several optimal solutions in X, we assume that the solution which is most profitable for the leader is chosen. That is the solution, which maximizes the value $p^T x^*$. Moreover, we assume that there is a point $(x, y) \in X$ with x being the k-dimensional all-zeroes vector. This simply means that the follower has a solution that does not give any revenue to the leader. Otherwise the revenue maximization problem would be unbounded which is obviously not an interesting case.

Before we can state our results for the new model, we need to introduce a number of technical notions. Given a feasible solution $(x, y) \in X$ of the follower, we call the value $\mathbf{1}^T x = \sum_{i=1}^{k} x_i$ the *mass* of (x, y). A *single-price* is a price vector p of the form $p = \lambda \mathbf{1}$ where λ is some real number. Slightly abusing notation, we sometimes call λ the single-price. Note that when the leader uses a single-price the revenue is simply the mass of the follower's solution times the single-price.

Let M be the maximum mass the follower buys if the leader sets all her prices to 0. Formally,

$$\begin{aligned} M := \max \quad & \mathbf{1}^T x \\ \text{s.t.} \quad & \exists y \in \mathbb{R}^{n-k} : \ (x, y) = \arg\min\{f(x', y') : (x', y') \in X\}. \end{aligned}$$

This value M exists since X is a compact set.

Consider, for example, the case where the follower seeks to buy a shortest s-t-path in a network. Then M is the maximum number of priceable edges of a shortest s-t-path in the network, when the priceable edges all have a price of 0 and thus can be bought for free by the follower.

Since X is a compact set, there exists a largest single-price at which the follower buys a non-zero mass from the leader. Let μ be the maximum mass the follower buys at this price. Consider again the case where the follower searches for a shortest s-t-path in a network. Then μ is the maximum number of priceable edges contained in a shortest s-t-path, under the largest single-price for which a shortest path exists that contains a priceable edge.

For all $m \in [0, M]$, let $\Delta(m)$ be the minimum price the follower has to pay if he buys a mass of at most m from the leader. More formally

$$\begin{aligned} \Delta(m) := \min \quad & f(x, y) \\ \text{s.t.} \quad & \mathbf{1}^t x \leq m \\ & (x, y) \in X, \end{aligned}$$

where $\mathbf{1}^t x$ is the mass bought by the follower. This minimum price $\Delta(m)$ exists, because X with the additional constraint of $\mathbf{1}^t x \leq m$ is again a compact set.

As observed by several authors (cf. [2, 5]), an upper bound on the optimum revenue is $R := \Delta(0) - \Delta(M)$. To see this, let r^* be the maximum revenue, and let (x^*, y^*) be the corresponding follower's solution. We have

$$r^* + \Delta(M) \leq r^* + \Delta(\mathbf{1}^T x^*) \leq r^* + f(x^*, y^*) \leq \Delta(0),$$

because $\Delta(0)$ is an upper bound on the objective value of the follower and Δ is non-increasing. We remark that R is indeed a tight upper bound, in the sense that there are examples of games where the maximum revenue equals R, e.g., the minimum spanning tree pricing problem described in the introduction.

As our first result shows, the maximum revenue of the leader is always reasonably close to R, unless the ratio M/μ is large. This is true even if the leader uses a single-price strategy.

▶ **Theorem 2.** *There is a single-price for the Stackelberg Pricing Game over X whose revenue is at least*

$$\frac{R}{1 + \ln\left(\frac{M}{\mu}\right)}.$$

This result extends previous work of Briest et al. [7] and Balcan et al. [2], who proved the above theorem in the combinatorial setting, i.e., for $X \subseteq \{0, 1\}^n$. We give the main idea of the proof of Theorem 2 in the following sketch and defer the full proof to the appendix.

Proof Sketch for Theorem 2. Our proof makes use of the following concept. For each $m \in (0, M]$, let $P(m)$ be the supremum of all single-prices for which the follower has an optimal solution with a mass of at least m from the leader. Using the compactness of X and the continuity of f one can prove that at the single-price of $P(m)$, the follower has an optimal solution of a mass of at least m. Thus, the supremum is indeed a maximum here.

It turns out that there are certain mass values which dictate the value of the function P. Let $T \subseteq (0, M]$ be the set of mass values t for which the follower does not have an optimal solution, at a single-price $P(t)$, with a mass more than t. The set T plays a key role in our proof, as the following claim indicates.

▶ **Claim 3.** *It holds that $\mu \in T$, $M \in T$ and $\mu = \min T$. Moreover, for all $m \in (0, M]$ it holds that $P(m) = \max_{t \in T \cap [m, M]} P(t)$.*

Recall that $\Delta(m)$ is defined as the price the follower has to pay for the non-priceable variables if he buys a mass of at most m from the leader. Similar to the proof of Briest et al. [7] for the combinatorial setting, we next show that the functions P and Δ are closely related. In our case, however, we have to deal with several difficulties that arise because we allow for non-discrete optimization problems of the follower.

Consider the lower convex hull H of the point set $\{(m, \Delta(m)) : 0 \leq m \leq M\}$. Let ∂H be the lower border of H, and let $\hat{\Delta} : [0, M] \to \mathbb{R}$ be the function for which $(m, \hat{\Delta}(m)) \in \partial H$ for all $m \in [0, M]$. We remark that, since $\hat{\Delta}$ is convex and decreasing,

$$D_-\hat{\Delta}(m) = \sup_{\ell<m} \frac{\hat{\Delta}(m) - \hat{\Delta}(\ell)}{m - \ell}, \text{ for each } m \in (0, M].$$

Here, D_- denotes the *lower left Dini derivative* of $\hat{\Delta}$. It is defined, for all $m \in (0, M]$, by

$$D_-\hat{\Delta}(m) = \liminf_{h\to 0^-} \frac{\hat{\Delta}(m) - \hat{\Delta}(m+h)}{h}.$$

As our main claim shows, the values of $P(m)$ and $D_-\hat{\Delta}(m)$ are essentially equal.

▶ **Claim 4.** *Except for a set of measure 0, it holds for all $m \in (0, M)$ that $D_-\hat{\Delta}(m) = -P(m)$.*

To establish Claim 4 is indeed the difficult part of the whole proof. We skip the details due to space constraints.

In the calculation that finishes the proof we make use of the inverse operation of the lower left Dini derivative, the so-called *lower left Dini integral*, denoted $(LD)\underline{\int}$. For more background we refer to the article of Hagood and Thomson [11].

We have

$$R = \Delta(0) - \Delta(M) = \hat{\Delta}(0) - \hat{\Delta}(M) = \lim_{\epsilon\to 0^+} \hat{\Delta}(\epsilon) - \hat{\Delta}(M) = \lim_{\epsilon\to 0^+} (\text{LD}) \underline{\int_\epsilon^M} -D_-\hat{\Delta}(m)\, dm.$$

Due to Claim 4, the integral

$$\lim_{\epsilon\to 0^+} (\text{LD}) \underline{\int_\epsilon^M} P(m)\, dm$$

is well defined and equals

$$\lim_{\epsilon\to 0^+} (\text{LD}) \underline{\int_\epsilon^M} - D_-\hat{\Delta}(m)\, dm.$$

Recall that, by Claim 3, $\mu = \min T$ and so $P(m) = P(\mu)$ for all $m \in (0, \mu]$. Hence,

$$\begin{aligned}\lim_{\epsilon\to 0^+} (\text{LD}) \underline{\int_\epsilon^M} P(m)\, dm &= \lim_{\epsilon\to 0^+} (\text{LD}) \underline{\int_\epsilon^\mu} P(m)\, dm + (\text{LD}) \underline{\int_\mu^M} P(m)\, dm \\ &= \mu \cdot P(\mu) + (\text{LD}) \underline{\int_\mu^M} P(m)\, dm.\end{aligned}$$

Let r be the maximum revenue achieved by the single-price strategy. Note that r is at least the revenue at the single-price $P(m)$, for each $m \in (0, M]$, which is in turn at least $m \cdot P(m)$. We thus have

$$\begin{aligned}\mu \cdot P(\mu) + (\text{LD}) \underline{\int_\mu^M} P(m)\, dm &= \mu \cdot P(\mu) + (\text{LD}) \underline{\int_\mu^M} \frac{m \cdot P(m)}{m}\, dm \\ &\leq r + (\text{LD}) \underline{\int_\mu^M} \frac{r}{m}\, dm \\ &= r + r \cdot (\ln(M) - \ln(\mu)) \\ &= r\left(1 + \ln\left(\frac{M}{\mu}\right)\right).\end{aligned}$$

This shows that $R \leq r(1 + \ln(M/\mu))$, as desired. ◀

Balcan et al. [2] and Briest et al. [7] extend their result to the situation when there are several followers.[1] The same approach works in our generalized model.

Assume there are ℓ followers and each follower has its own optimization problem. Formally, the i-th follower minimizes his objective function $p^T x^i + f_i(x^i, y^i)$ where (x^i, y^i) belongs to the set $X_i \subseteq \mathbb{R}^k_{\geq 0} \times \mathbb{R}^{n-k}_{\geq 0}$ of his feasible solutions, $i = 1, \ldots, \ell$. The pricing vector p appears in the objective function of every follower and is again set by the leader in order to maximize her revenue $\sum_{i=1}^{\ell} p^T x^i$. The difficulty here is that, while each follower has an individual optimization problem, the leader can set only one price vector for all followers at once. There is, however, a canonical way of reducing the pricing game to the case of a single follower. To this end, we consider the pricing game with respect to follower i only, and let μ_i (resp. M_i) be the minimum non-zero mass (resp. the maximum mass) bought by follower i. Moreover, let R_i be the upper bound on the revenue with respect to follower i.

▶ **Corollary 5.** *There is a single-price for the Stackelberg Pricing Game with ℓ followers whose revenue is at least*

$$\frac{\sum_{i=1}^{\ell} R_i}{1 + \ln\left(\frac{\sum_{i=1}^{\ell} M_i}{\min_{i=1}^{\ell} \mu_i}\right)}.$$

To see this, consider a single follower with the feasible subset $X = X_1 \times X_2 \times \ldots \times X_\ell$. It is easy to see that we have $M = \sum_{i=1}^{\ell} M_i$ and $R = \sum_{i=1}^{\ell} R_i$ in this game. Moreover, the smallest non-zero mass μ bought by the newly defined single follower is at least the minimum smallest non-zero mass bought by one of the ℓ followers, that is $\min_{i=1}^{\ell} \mu_i$. Now applying Theorem 2 to the single follower yields Corollary 5.

Theorem 2 is tight in the following sense.

▶ **Proposition 6.** *There are Stackelberg Pricing Games of arbitrarily large R and M in which the optimum revenue equals*

$$(1 + o(1)) \cdot \frac{R}{1 + \ln\left(\frac{M}{\mu}\right)}.$$

This holds true even for games in which the follower minimizes a linear objective function of the form $p^T x + c^T y$ over a uniform matroid.

Note that, in the above statement, every possible pricing is considered and not just single-price strategies. In other words, the lower bound in Theorem 2 is tight not only for single-price strategies, but for arbitrary pricings. So far, it was known that there are combinatorial pricing games where the optimum revenue is in $O(R/\log k)$, where k is the number of priceable elements (cf. [5]). The merit of Proposition 6 is that it shows tightness of Theorem 2 up to a factor of $1 + o(1)$, which is best possible. This fact, and the construction given in the proof of Proposition 6, enable us to prove the following hardness result.

▶ **Theorem 7.** *Fix a sufficiently small rational number $\epsilon > 0$, and consider a Stackelberg Pricing Game where the follower minimizes an objective function of the form $p^T x + c^T y$ over a matroid. It is strongly NP-hard to decide whether there is some pricing of revenue at least*

$$(1 + \epsilon) \cdot \frac{R}{1 + \ln\left(\frac{M}{\mu}\right)}.$$

[1] In this paper we consider the case of *unlimited supply*, meaning that the followers buy their favorite solution independently of each other.

Due to space constraints, the proof of Theorem 7 is deferred to the appendix. In the statement of the above theorem, we assume that the matroid is given by its ground set and a membership oracle. Thus, μ, M, and R can be computed in polynomial time. Moreover, it will be clear from the proof that the natural logarithm can be computed in polynomial time with the precision needed to decide the problem.

Let us remark that the assumption of ϵ being *sufficiently small* is merely technical, and can be dropped by giving a more careful hardness reduction. The message of the above theorem is, however, that there is a sharp contrast between the revenue guaranteed by the simple single-price strategy given in Theorem 2 and anything more than that.

3 Parameterized complexity of Stackelberg Pricing Games with few priceable objects

In this section we study the parameterized complexity of Stackelberg Pricing Games when parameterized by the number of priceable objects. Intuitively, this addresses the question of whether there are improved algorithms for the case that only few objects are priceable. We give a general positive result for the combinatorial model. Our main result in this part, however, is a hardness proof for the linear programming case; we begin with the latter.

In the LP-PRICING problem there is a linear program over which the follower minimizes. The leader may choose the price, i.e. target function coefficient, of k specified variables. Her revenue is determined by the corresponding (weighted) sum over these variables.

LP-PRICING
Input: A linear program with k priceable variables and $\lambda \in \mathbb{Q}$.
Question: Is there a price vector whose revenue is at least λ?

We prove that this problem is at least as hard to solve as the parameterized k-clique problem. The hardness proof creates linear programs with only non-negative variables and non-negative target function over which the follower seeks to minimize. As such it proves hardness also for our more general model parameterized by number of priceable variables.

▶ **Theorem 8.** LP-PRICING *is* W[1]*-hard when parameterized by the number k of priceable variables.*

The theorem is proven by a reduction from the well-known (parameterized) MULTI-COLORED CLIQUE problem. Therein, we are given a k-partite graph G (or, equivalently, a properly k-colored graph) and have to determine whether G contains a clique on k vertices; the problem is W[1]-hard with respect to parameter k. Thus, unless FPT = W[1], there is no algorithm running in time $f(k)n^c$ for instances of size n. Moreover, under the Exponential-Time Hypothesis [13] the reduction implies that there is no $O(n^{o(k)})$ time algorithm for LP-PRICING. In instances created by the reduction the leader can effectively enforce the choice of the k clique vertices by setting appropriate prices for k variables; the remaining variables are used to verify the choice and a certain revenue threshold can only be attained if there is indeed a k-clique. The behavior of these k priceable variables is quite similar to integer variables, as they can be shown to only take specific values from a finite set in solutions meeting the threshold (one value corresponding to each vertex of G). This arguably gives our parameterized LP PRICING problem some similarity to the MIXED ILP FEASIBILITY problem parameterized by the number of integer variables. Interestingly, the latter problem is FPT due to a classic result of Lenstra [12]. Due to space constraints we will restrict ourselves to an informal description of the reduction complemented by some intuition.

Proof Sketch for Theorem 8. We give a parameterized reduction from the W[1]-hard MULTICOLORED CLIQUE(k) problem. Therein, we are given a k-partite graph $G = (V_1, \dots, V_k, E)$ and have to determine whether it contains a clique of size k, i.e., a clique containing exactly one vertex from each partite set. The reduction is polynomial-time computable and creates an instance of LP-PRICING with $k + 1$ priceable variables, proving W[1]-hardness of LP-PRICING.

Much effort in the design of the linear program and the correctness proof goes into setting up constraints on k pairs of variables (x_i, y_i) and forcing each pair to take a value in $\{(p_1, q_1), \dots, (p_n, q_n)\}$; each x_i is priceable and the y_i have fixed prices. We have $p_1 > p_2 > \dots > p_n$ and $q_1 < q_2 < \dots < q_n$ with the high-level idea that changing the price of x_i will make a particular point (p_j, q_j) optimal. Each choice of $(x_i, y_i) = (p_j, q_j)$ corresponds to the selection of one vertex (i, v_i) from the ith partite set V_i of G. Additionally, there are variables $z_{i,j} \geq |y_i - q_j|$ that serve as indicators for whether $y_i = q_j$, and clique-testing constraints on pairs of variables $z_{i,u}, z_{j,v}$ for every non-edge $\{(i, u), (j, v)\}$ of G to prevent choosing both (i, u) and (j, v) for the clique. Notably, there is a single slack variable y_0 present in all clique-testing constraints that is connected to a priceable variable x_0 in such a way that the maximum revenue c_0 for x_0 is only possible if the slack y_0 is not needed, i.e., if all clique-testing constraints are active.

The reader will have noticed that there are no constraints that we could pose in a linear program that would enforce $(x_i, y_i) \in \{(p_1, q_1), \dots, (p_n, q_n)\}$. Instead, the points (p_j, q_j) are the extremal points (when projected to two dimensions) of certain constraints on each pair x_i and y_i; we call these the *core constraints*. There are intended prices $r_1, \dots, r_n$ with the goal of showing that setting the leader price d_i of x_i to r_j leads to $(x_i, y_i) = (p_j, q_j)$ being uniquely optimal. This would probably be easier if we could restrict to $d_i \in \{r_1, \dots, r_n\}$, but we are not allowed to do so. Instead, the strategy is roughly as follows:

1. Consecutive pairs of points (p_j, q_j) and (p_{j+1}, q_{j+1}) are chosen in such a way that for $d_i = r_j$ all choices of (x_i, y_i) on the line segment defined by the two points give the same follower cost including updates to indicator variables $z_{i,j}$. (This is rather helpful for replacement arguments, which we use to disprove solutions not following intended behavior, but does not cover the variable y_0.) The leader payoff in this case is highest for $x_i = p_j$ as $p_j > p_{j+1}$, and we have $r_j := \frac{1}{p_j}$ so that the payoff is exactly 1 in this case.
2. A critical issue is that we must make sure that the leader cannot make a profit larger than 1 with any variable x_i since that may be more beneficial than attempting to gain the revenue for having the clique-testing constraints fulfilled without increasing the slack variable y_0 above 0.
3. A significant part of the proof hence goes towards a technical claim that allows to rule out both profit greater than one for any variable x_i and the case that $p_s > x_i > p_{s+1}$, i.e., that x_i lies between two intended coordinates. To this end, the core constraints on pairs (x_i, y_i) contain varying numbers of slack variables $w_{i,1}, \dots, w_{i,n}$ of fixed prices $r_1, \dots, r_n$. These are placed in such a way that they provide alternative ways of satisfying the core constraints when $d_i \notin \{r_1, \dots, r_n\}$, without keeping the follower from actually paying the full price of x_i in other cases.

We remark that the correctness proof of course also requires the reverse direction of ensuring leader payoff at least $k + c_0$ if G has a clique of size k. Here we are in the easier situation of being able to select leader prices and fixing a suggested solution. We then prove feasibility, which is straightforward, and optimality, which requires combining most of the constraints into a lower bound on the optimal follower cost that matches that of our suggested solution. ◀

Theorem 8 is in sharp contrast to the combinatorial setting, where under mild assumptions one can see the problem to be fixed-parameter tractable. Here we assume that $X \subseteq \{0,1\}^n$ but put no further restriction on the follower's objective function $f : X \to \mathbb{R}$. In particular, this model covers the classical setting where each item has a fixed cost and if the follower buys a set S of items, he has to pay the sum of the fixed costs of the elements of S, plus the leader's price of the bought elements.

▶ **Theorem 9.** *Assume that $X \subseteq \{0,1\}^n$, and that we are given a polynomial-time algorithm to compute an optimal solution of the follower for given leader prices $p \in \mathbb{R}^k$. Then the computation of optimal prices and optimal leader revenue is fixed-parameter tractable, with running time $\mathcal{O}(2^k n^c)$, when parameterized by the number of priceable items.*

In the above statement we make the natural assumption that the input size is at least $n + \text{size}(f)$, where $\text{size}(f)$ denotes the maximum length of the binary encoding of any value f can take.

Proof Sketch for Theorem 9. Due to space constraints the proofs of the claims in this section are deferred to the appendix. We need the following proposition by Briest et al. [7]. It says that if the leader wants to force the follower to pick a certain solution, she can compute suitable prices in polynomial time. We state their result in a slightly more general fashion than in the original paper. Indeed, Briest et al. were only concerned with the case when f is additive but the proof did not make use of the additivity of f at all.

▶ **Proposition 10** (Briest et al. [7])**.** *Given a vector $z \in X$, one can compute an optimal price vector p such that z is an optimal solution of the follower with respect to p, or decide that such a price vector does not exist, in polynomial time.*

Our algorithm works as follows. For each vector $x \in \{0,1\}^k$ we compute a price vector p_x, if exists, such that

(a) there is some $y \in \{0,1\}^{n-k}$ such that the vector (x,y) is an optimal solution of the follower with respect to the price vector p_x,

(b) subject to (a) the revenue $p_x^T x$ is maximum.

When this procedure is finished we choose the vector $\hat{x} \in \{0,1\}^k$ with maximum value of ${p_{\hat{x}}}^T \hat{x}$, and output the price vector $p_{\hat{x}}$. As the next claim shows, this price vector is the optimum solution.

▶ **Claim 11.** *The output $p_{\hat{x}}$ is an optimal price vector for the leader.*

In the remainder of the proof we show how to compute p_x for a fixed candidate vector $x \in \{0,1\}^k$. First we aim to find a vector $y_x \in \{0,1\}^{n-k}$ such that $(x, y_x) \in X$ and, subject to this, $f(x, y_x)$ is minimum. Note that possibly such a vector y_x does not exist. To find a vector y_x, or decide that none exists, we define a price vector p by setting

$$p_i = \begin{cases} -M & \text{if } x_i = 1, \\ M & \text{if } x_i = 0, \end{cases}$$

for all $i \in [k]$. Here, M is a number that is large enough to ensure that

$$\text{for all } (x', y'), (x'', y'') \in X, \text{ it holds that } f(x', y') - f(x'', y'') < M. \tag{1}$$

As both values $|f(x', y')|$ and $|f(x'', y'')|$ are bounded by $2^{\text{size}(f)}$, we may simply put $M = 2^{\text{size}(f)+1} + 1$.

Now we compute an optimal solution of the follower with respect to the price vector p, say (x^*, y^*), using the assumed polynomial-time algorithm.

▶ **Claim 12.** *If for some $y \in \{0,1\}^{n-k}$ it holds that $(x,y) \in X$, then $x^* = x$.*

If $x^* \neq x$, Claim 12 implies that there is no price vector satisfying (a). Thus, we may safely abort the process and go over to the next candidate vector x. Otherwise if $x^* = x$, we put $y_x = y^*$.

▶ **Claim 13.** *$f(x,y_x) \leq f(x,y)$ holds for all $y \in \{0,1\}^{n-k}$ with $(x,y) \in X$.*

Next, we use Proposition 10 to compute a price vector p_x such that the vector (x,y_x) is optimal for the follower and, subject to that, ${p_x}^T x$ is maximum. Note that this price vector does exist since, e.g., the vector p is a feasible price vector. By construction, p_x has the property (a). So far, we only know that ${p_x}^T x$ is maximum subject to the condition that under the price vector p_x the vector (x,y_x) is an optimal solution of the follower.

▶ **Claim 14.** *Subject to (a), the revenue $p_x^T x$ is maximum.*

As the running time of the whole algorithm is $O(2^k \cdot (n + \text{size}(f))^{O(1)})$, the proof is complete. ◀

4 Conclusion and future work

The basis for the first part of this paper were the results of Briest et al. [7] and Balcan et al. [2] who gave a lower bound on the optimal revenue in Stackelberg Network Pricing Games. We proved that this bound carries over to a much more general setting, where, basically, the follower minimizes a continuous function over a compact set of points. This model captures important settings that are not covered by the classical combinatorial model. For example, the case when the follower is minimizing a linear program, e.g., a minimum cost flow problem.

The proven lower bound also holds if a single-price strategy is applied, and it is tight up to a factor of $(1 + o(1))$. Moreover, we used this tightness example to show that it is strongly NP-hard to decide whether the revenue of an optimal pricing exceeds the lower bound by an arbitrarily small linear factor.

In the second part of the paper we studied the parameterized complexity of the revenue maximization problem. It turned out that in the combinatorial setting (i.e., when the follower only has 0/1-valued solutions) there is an elegant FPT algorithm. Once we leave this regime, however, things become more difficult. Indeed, if the follower has an optimization problem in the form of a linear program, the revenue maximization problem becomes W[1]-hard and is thus most likely not FPT.

Several central questions remain. Most importantly, one should consider multiple-follower scenarios. An intriguing model is when the particular resources have a limited supply. In the combinatorial setting, a limited supply means that every item to be sold is available only a limited number of times. Now the followers come one by one, in a certain order, and buy according to their preferences and the prices set by the leader. Balcan et al. [2] prove a tight lower bound on the revenue obtained by the single-price strategy. In the non-combinatorial model, the limited supply might be translated to a constraint of the form $(x,y) \leq s$, where $s \in \mathbb{R}^n_{\geq 0}$ is a fixed vector that is added to the usual constraint $(x,y) \in X$ in the optimization problem of the followers.

References

1 Mourad Baïou and Francisco Barahona. Stackelberg Bipartite Vertex Cover and the Preflow Algorithm. to appear in Algorithmica, 2016.

2 Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. Item Pricing for Revenue Maximization. In *Proceedings 9th ACM Conference on Electronic Commerce (EC-2008), Chicago, IL, USA, June 8-12, 2008*, pages 50–59, 2008. doi:10.1145/1386790.1386802.

3 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Specializations and Generalizations of the Stackelberg Minimum Spanning Tree Game. *Theoretical Computer Science*, 562:643–657, 2015.

4 Davide Bilò, Luciano Gualà, Guido Proietti, and Peter Widmayer. Computational Aspects of a 2-player Stackelberg Shortest Paths Tree Game. In *Internet and Network Economics*, pages 251–262. Springer, 2008.

5 Patrick Briest, Parinya Chalermsook, Sanjeev Khanna, Bundit Laekhanukit, and Danupon Nanongkai. Improved Hardness of Approximation for Stackelberg Shortest-Path Pricing. In *Internet and Network Economics*, pages 444–454. Springer, 2010.

6 Patrick Briest, Luciano Gualà, Martin Hoefer, and Carmine Ventre. On stackelberg pricing with computationally bounded customers. *Networks*, 60(1):31–44, 2012. doi:10.1002/net.20457.

7 Patrick Briest, Martin Hoefer, and Piotr Krysta. Stackelberg Network Pricing Games. *Algorithmica*, 62(3-4):733–753, 2012. doi:10.1007/s00453-010-9480-3.

8 Sergio Cabello. Stackelberg Shortest Path Tree Game, Revisited. *arXiv preprint arXiv:1207.2317*, 2012.

9 J. Cardinal, E. D. Demaine, S. Fiorini, G. Joret, S. Langerman, I. Newman, and O. Weimann. The Stackelberg Minimum Spanning Tree Game. *Algorithmica*, 59:129–144, 2011.

10 Jean Cardinal, Erik D Demaine, Samuel Fiorini, Gwenaël Joret, Ilan Newman, and Oren Weimann. The Stackelberg Minimum Spanning Tree Game on Planar and Bounded-Treewidth Graphs. *Journal of combinatorial optimization*, 25(1):19–46, 2013.

11 J. W. Hagood and B. S. Thomson. Recovering a Function from a Dini Derivative. *The American Mathematical Monthly*, 113:34–46, 2006.

12 Jr. H. W. Lenstra. Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

13 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

14 Gwenaël Joret. Stackelberg Network Pricing is Hard to Approximate. *Networks*, 57(2):117–120, 2011.

15 M. Labbé, P. Marcotte, and G. Savard. A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing. *Management Science*, 44:1608–1622, 1998.

16 S. Roche, G. Savard, and P. Marcotte. An approximation algorithm for Stackelberg network pricing. *Networks*, 46:57–67, 2005.

17 S. van Hoesel. An overview of Stackelberg pricing in networks. *European Journal of Operational Research*, 189:1393–1402, 2008.

18 H. von Stackelberg. *Marktform und Gleichgewicht*. Springer, 1934.

Online Market Intermediation*†

Yiannis Giannakopoulos[1], Elias Koutsoupias[2], and Philip Lazos[3]

1 **Department of Informatics, TU Munich, Munich, Germany**
giannako@in.tum.de
2 **Department of Computer Science, University of Oxford, Oxford, UK**
elias@cs.ox.ac.uk
3 **Department of Computer Science, University of Oxford, Oxford, UK**
filippos.lazos@cs.ox.ac.uk

Abstract

We study a dynamic market setting where an intermediary interacts with an unknown large sequence of agents that can be either sellers or buyers: their identities, as well as the sequence length n, are decided in an adversarial, online way. Each agent is interested in trading a single item, and all items in the market are identical. The intermediary has some prior, incomplete knowledge of the agents' values for the items: all seller values are independently drawn from the same distribution F_S, and all buyer values from F_B. The two distributions may differ, and we make common regularity assumptions, namely that F_B is MHR and F_S is log-concave.

We focus on online, posted-price mechanisms, and analyse two objectives: that of maximizing the intermediary's profit and that of maximizing the social welfare, under a competitive analysis benchmark. First, on the negative side, for general agent sequences we prove tight competitive ratios of $\Theta(\sqrt{n})$ and $\Theta(\ln n)$, respectively for the two objectives. On the other hand, under the extra assumption that the intermediary knows some bound α on the ratio between the number of sellers and buyers, we design asymptotically optimal online mechanisms with competitive ratios of $1 + o(1)$ and 4, respectively. Additionally, we study the model where the number of items that can be stored in stock throughout the execution is bounded, in which case the competitive ratio for the profit is improved to $O(\ln n)$.

1998 ACM Subject Classification J.4 [Social and Behavioral Sciences] Economics

Keywords and phrases optimal auctions, bilateral trade, sequential auctions, online algorithms, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.47

1 Introduction

The design and analysis of electronic markets is of central importance in algorithmic game theory. Of particular interest are trading settings, where multiple parties such as buyers, sellers, and intermediaries exchange goods and money. Typical examples are markets for trading stocks, commodities, and derivatives: sellers and buyers where each one trades a single item and one intermediary for facilitating the transactions. However, the well-understood cases are comparatively quite modest. The very special case of one seller, and one buyer was thoroughly studied by Myerson and Satterthwaite [26] in their seminal paper; they provided a beautiful characterization of many significant properties a mechanism might have, along

* A full version of this paper can be found in [18], http://arxiv.org/abs/1703.09279.
† This work was kindly supported by the ERC Advanced Grant 321171 (ALGAME), ERC Advanced Grant 691672 (APEG) and by the EPSRC (award reference 1493310).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 47; pp. 47:1–47:14


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

with an impossibility theorem showing that it cannot possess them all. The paper also dealt with the case where a broker provides assistance by making two potential trades, one with each agent, while also trying to maximize his profit. This was extended in [15] to multiple sellers and buyers that are all immediately present in an offline manner.

Our work considers a similar setting, but with a key difference: the buyers and sellers appear one-by-one, in a dynamic way. It is natural to study this question in the incomplete information setting in which the intermediary, whose objective is to maximize either profit or welfare, does not know the sequence of buyers and sellers in advance. The framework that we employ to study the question is the standard worst-case analysis of online algorithms whose goal is to do as well as possible in the face of a powerful adversary which tries to embarrass them.

We are not the first to apply techniques from online algorithms to quantify uncertainty in markets: the closest work to ours would be by Blum et al. [8] who consider buyers and sellers trading identical items. In their setting, motivated mostly from a financial standpoint, buyers and sellers arrived in an online manner, with their bids appearing to the trader and expiring after some time. The trader would have to match prospective buyers and sellers to facilitate trade. Among a plethora of interesting results, the trader's profit maximization problem was studied using competitive analysis and techniques from online weighted matchings. The key difference in our setting is that buyers and sellers do not overlap: whenever a seller appears, the intermediary has to decide whether or not to attempt to buy the item, without having a buyer ready to go. Instead, the intermediary stores the item to sell it at a later time. We believe this variation is able to capture "slower" markets, like online marketplaces similar to Amazon or AliExpress (or even regular retail stores), where uncertainty stems from not knowing how large a stock of items to buy, in expectation of the buyers to come.

1.1 Our Results

Our aim is to study this dynamic market setting, where an intermediary faces a sequence of potential buyers and sellers in an online fashion. The goal of the intermediary is to maximize his profit, or society's welfare, by buying from the sellers and selling to buyers. We take a Bayesian approach to their utilities but use competitive analysis for their arrivals: the main difficulty stems from the unknown (and adversarially chosen) sequence of agents. Further particulars and notation is discussed in Section 2. All the online algorithms we design are posted price, which are simple, robust and strongly truthful.

First, in Section 4 we study the case of arbitrary sequences of buyers and sellers and show that the competitive ratio—the ratio of the optimal offline profit over the profit obtained by the online algorithm—is $\Theta(\sqrt{n})$, where n is the total number of buyers and sellers. We also study the social welfare objective, where the goal is to maximize the total utility of all participants, including the sellers, the buyers and the intermediary The competitive ratio here is $\Theta(\log n)$. All these results are achieved via common regularity assumptions on the distributions of the agent values (see Section 3), which we also prove to be necessary, by providing arbitrarily bad competitive ratios in the case they are dropped (Theorem 7).

To overcome the above pessimistic results, we next study in Section 5 the setting where both the online and offline algorithms have a limited stock, i.e. at no point in time can they hold more than K items. In this model, the competitive ratio is improved to $\Theta(K \log n)$, asymptotically matching that of welfare. Finally, we also propose a way to restrict the input sequence, by introducing in Section 6 the notion of α-balanced streams, where at every prefix of the stream the ratio of the number of sellers to buyers has to be at least α. Under this condition we are able to bring down the competitive ratios for both objectives to constants. In

particular, the online posted-price mechanism that we use for profit maximization, and which is derived by a fractional relaxation of the optimal offline profit, achieves an asymptotically optimal ratio of $1 + o(1)$. A similar mechanism is 4-competitive for the welfare objective.

All omitted proofs can be found in the full version of the paper [18].

1.2 Prior Work

Our work is grounded on a string of fruitful research in mechanism design. The main topics that are close to our effort are bilateral trading, trading markets and sequential (online) auctions.

The first step in bilateral trading and mechanism design was made by Myerson and Satterthwaite [26] who proved their famous impossibility result, even for the case of one buyer and one seller. The case for profit maximization was extended to many buyers and sellers, each trading a single identical item, in [15]. Some of the assumptions in our model are based in these two works. The impossibility result in [26], among other difficulties, slowly vanishes for larger markets as was shown by McAffee [25]. There is still active progress being made on this intriguing setting, concentrating on simple mechanisms that provide good approximations either to welfare while staying budget balanced and individually rational [9, 11] or to profit [27]. Other recent developments include a hardness result for computing optimal prices [17] and constant efficiency approximation with strong budget balance [14].

Sequential auctions have also produced a collection of interesting results, either extending the ideas of simple approximate mechanisms instead of more complex, theoretically optimal ones or dealing with entirely new settings. Prominent examples that compare the revenue (or welfare) generated by simple, posted-price sequential auctions to the optimal, proving good approximations in certain cases, are [10] for single-item revenue, [13, 29] for matroid constraints (and some multi-dimensional settings) and [16] for combinatorial auctions. There have been many approaches that apply competitive (worst-case) analysis to mechanism design. The analysis of auctions with unlimited supply is explored in [5, 7] where near optimal algorithms are developed using techniques inspired from no-regret learning. There is also a deep connection between secretary problems and online sequential auctions [21, 20, 3]. Hajiaghayi et al. utilized techniques such as prophet inequalities for unknown market size with distributional assumptions in [22]. A comprehensive exposition of online mechanism design by Parkes can be found in [28].

There are also positive results in online auctions when the valuation distribution is unknown (but usually known to be restricted in some way, having bounded support or being monotone hazard-rate etc). Babaioff et al. explored the case of selling a single item to multiple i.i.d. buyers in [1]. The case of k items in a similar setting was studied in [2], while the case of unlimited items (digital goods auctions) in [23] and [24]. Budget constraints where also introduced in [4], where a procurement auction was the focus.

2 Preliminaries and Notation

The input is a finite string $\sigma \in \{S, B\}^*$ of buyers (B) and sellers (S). The online algorithm has no knowledge of $\sigma(t)$, i.e. whether $\sigma(t) = S$ or $\sigma(t) = B$, before step t. Also, it doesn't know the length $n(\sigma)$ of σ. Denote $n_S(\sigma)$, $n_B(\sigma)$ the number of sellers and buyers, respectively, in σ, and let $N_S(\sigma)$, $N_B(\sigma)$ be the corresponding set of indices, i.e. $N_S(\sigma) = \{t \mid \sigma(t) = S\}$ and $N_B(\sigma) = \{t \mid \sigma(t) = B\}$. Let $N(\sigma) = N_S(\sigma) \cup N_B(\sigma) = \{1, 2, \ldots, n(\sigma)\}$. In the above notation we will often drop the σ if it is clear which input stream we are referring to.

The values of the sellers are drawn i.i.d. from a probability distribution (with cdf) F_S and these of buyers i.i.d. from a distribution F_B, both supported over intervals of nonnegative reals. We denote the random variable of the value of the t-th agent with X_t. We assume that distributions F_S and F_B are continuous, with bounded expectations μ_S and μ_B, and have (well-defined) density functions f_S and f_B, respectively. It will also be useful to denote by X_S a random variable drawn from distribution F_S, and similarly $X_B \sim F_B$, and for any random variable Y and positive integer m use $Y^{(m)}$ to represent the maximum order statistic out of m i.i.d. draws from the same distribution as Y. We will also use the shortcut notation $\mu^{(m)} = \mathbb{E}[Y^{(m)}]$.

We study *posted-price* online algorithms that upon seeing the identity of the t-th agent (whether she is a seller or a buyer), offer a price p_t. We buy one unit of the item from sellers that accept our price (i.e. if $\sigma(t) = S$ and $X_t \leq p_t$) and pay them that price, and we sell to buyers that accept our price (i.e. if $\sigma(t) = B$ and $X_t \geq p_t$), given stock availability (see below), and collect from them that price. So, a price p_{t+1} can only depend on $\sigma(1), \ldots, \sigma(t+1)$ and the result of the comparison $X_i \leq p_i$ in all previous steps $i = 1, 2, \ldots, t$. Let K_t denote the available stock at the beginning of the t-th step, i.e. $K_1 = 0$ and

$$K_{t+1} = \begin{cases} K_t + 1, & \text{if } \sigma(t) = S \ \wedge \ X_t \leq p_t \\ K_t - 1, & \text{if } \sigma(t) = B \ \wedge \ K_t \neq 0 \ \wedge \ X_t \geq p_t \\ K_t, & \text{otherwise.} \end{cases}$$

Then, the set of sellers from whom we bought items during the algorithm's execution is $I_S = \{t \in N_S \mid X_t \leq p_t\}$ and the set of buyers we sold to is $I_B = \{t \in N_B \mid X_t \geq p_t \wedge K_t \neq 0\}$. Notice that these are random variables, depending on the actual realizations of the agent values X_t.

The total *profit* that the intermediary deploying an algorithm A makes throughout the execution on an input stream σ, is the amount he manages to collect from the buyers via successful sales, minus the amount he spent in order to maintain stock availability from the sellers, that is

$$\mathcal{R}(A, \sigma) = \mathbb{E}\left[\sum_{t \in I_B} p_t - \sum_{t \in I_S} p_t\right].$$

The social *welfare* of algorithm A is the sum of valuations that all participants achieve throughout the entire execution. That is, a seller at position t of the stream has a value of X_t if she keeps her item, or a value of p_t if she sold the item to the intermediary; a buyer has a value of $X_t - p_t$ if she managed to buy an item, since the item has a value of X_t and he spent p_t to buy it, or 0 otherwise. And the intermediary, has a value of $\mathcal{R}(A)$ plus the value of the items that he didn't manage to sell in the end and which are now left in his stock. Putting everything together and performing the occurring cancellations, this results in the welfare to be expressed simply as the sum of the values of the sellers that kept their items plus the sum of the values of the buyers that bought an item, i.e.

$$\mathcal{W}(A, \sigma) = \mathbb{E}\left[\sum_{t \in N_S \setminus I_S} X_t + \sum_{t \in I_B} X_t\right]. \tag{1}$$

We use *competitive analysis*, the standard benchmark for online algorithms (see e.g. [12]), in order to quantify the performance of an online algorithm A: we compare it to that of an unrealistic, offline optimal algorithm OPT has access to the entire stream σ in advance.

Then, we say that A is $\rho(n)$-competitive with respect to welfare, if for any feasible input sequence of agents σ with length n and distributions F_S, F_B for the agent values, it is $\mathcal{W}(\text{OPT}, \sigma) \leq \rho(n) \cdot \mathcal{W}(A, \sigma)$. Notice how we allow the competitive ratio $\rho(n)$ to explicitly depend on the input's length, so that we can perform asymptotic analysis as $\mathcal{W}(\text{OPT}, \sigma)$ and n tend to infinity. It is common in competitive analysis to allow for an additional constant in the right hand side of the above expression, that does not depend in the input, and which intuitively can capture some initial configuration disadvantage of the online algorithm. We do that for the case of the profit objective, as this constant will have a very natural interpretation: you can think of it as the maximum amount of deficit on which an online algorithm can run at any point in time, since an adversary can always stop the execution at any time he wishes. Given that interpretation, it makes sense to allow for this constant to depend on seller distribution F_S, since even when we face a single seller at the first step we expect to spend an amount that depends on the realization of her value. Thus, we will say that an online algorithm is $\rho(n)$-competitive with respect to profit, if for any input sequence of agents σ and any probability priors F_S, F_B,

$$\mathcal{R}(\text{OPT}, \sigma) \leq \rho(n) \cdot \mathcal{R}(A, \sigma) + O(\mu_S). \tag{2}$$

3 Distributional Assumptions

Throughout most of the paper we will make some assumptions on the distributions F_B, F_S from which the buyer and seller values are drawn. In particular, we will assume that F_B has *monotone hazard rate (MHR)*, i.e. $\log(1 - F_B(x))$ is concave, and that F_S is *log-concave*, i.e. $\log F_S(x)$ is concave. For convenience, we will collectively refer to both the above constraints as *regularity assumptions*. These conditions are rather standard in the optimal auctions literature, and they encompass a large class of natural of distributions including e.g. exponential, uniform and normal ones. Notice that distributions that satisfy the above conditions also fulfil the regularity requirements introduced in the seminal paper Myerson and Satterthwaite [26] for the single-shot, one buyer and one seller setting of bilateral trade, namely that $x + \frac{F_S(x)}{f_S(x)}$ and $x - \frac{1-F_B(x)}{f_B(x)}$ are both increasing functions. Finally, we must mention that such regularity assumptions are necessary, in the sense that dropping them would result in arbitrarily bad lower bounds for the competitive ratios of our objectives, as it is demonstrated by Theorem 7.

The following two lemmas demonstrate some key properties of distributions satisfying our regularity assumptions and which will be very useful in our subsequent analysis:

▶ **Theorem 1.** *For any random variable Y drawn from an MHR distribution with bounded expectation μ and standard deviation s,*

1. $\Pr[Y \geq y] \geq \frac{1}{e}$ *for any* $y \leq \mu$
2. $\Pr[Y \geq y] < \frac{1}{e}$ *for any* $y > 2\mu$
3. $\mathbb{E}[Y^{(m)}] \leq H_m \cdot \mu$, *where H_m is the m-th harmonic number.*
4. $s \leq \mu$

Proof. A proof of Property 1 can be found in [6, Theorem 3.8], of Property 2 in [6, Corollary 3.10], and of Property 3 in [1, Lemma 13]. For Property 4, from [19, Lemma 2] we know that $\mathbb{E}\left[Y^2\right] \leq 2\mu^2$, so $s^2 = \mathbb{E}\left[Y^2\right] - \mu^2 \leq \mu^2$. ◀

▶ **Lemma 2.** *For any distribution over $[0, \infty)$ with log-concave cdf F and expectation μ,*

$$x \leq e\mu F(x) \qquad \text{for any } x \leq \mu.$$

Finally, we prove the following property bounding the sum of maximum order statistics of a distribution, that holds for general (not necessarily MHR) distributions and might be of independent interest:

▶ **Lemma 3.** *The expected average of the k-th highest out of m independent draws from a probability distribution with expectation μ and standard deviation s can be at most $\mu + 2\sqrt{\frac{m}{k}}s$.*

4 General Setting

We start by studying the general setting where no additional assumptions are enforced on the structure of the input sequence. The adversary is free to arbitrarily choose the identities of the agents.

4.1 Welfare

▶ **Theorem 4.** *Under our regularity assumptions*[1]*, the online auction that posts to any seller and buyer the median of their distribution is $O(\ln n)$-competitive with respect to welfare. This bound is tight.*

Proof. We split the proof of the theorem in two more general lemmas below, corresponding to upper and lower bounds. Then, the upper bound for our case follows easily from Lemma 5 by using constants $c_1 = c_2 = 2$, and taking into consideration that, from Property 3 of Theorem 1, the ratio of the maximum order statistic for the MHR distribution F_B is upper bounded by $r_B(m) \leq H_m \leq O(\ln m)$. For the lower bound, it is enough to observe that this ratio is attained by an exponential distribution, which is MHR.

▶ **Lemma 5.** *For any choice of constants $c_1, c_2 > 1$, the following fixed-price online auction has a competitive ratio of at most $\max\left\{\frac{c_1}{c_1-1}, c_1 c_2 \cdot r_B(n_B)\right\}$ with respect to welfare, where n_B is the number of buyers, and $r_B(m) = \mu_B^{(m)}/\mu_B$ is the ratio between the m-maximum-order statistic and the expectation of the buyer value distribution.*

- *Post to all sellers price $q = F_S^{-1}\left(\frac{1}{c_1}\right)$.*
- *Post to all buyers price $p = F_B^{-1}\left(\frac{c_2-1}{c_2}\right)$.*

Proof. Let A denote our online algorithm and OPT an offline algorithm with optimal expected welfare. Fix an input stream σ. Looking at (1), the maximum welfare that OPT can get from the sellers is at most $\mathbb{E}\left[\sum_{t\in N_S} X_t\right] = n_s\mu_S$, while from the buyers at most $\mathbb{E}\left[|I_B| \cdot X_B^{(n_B)}\right] \leq \kappa\,\mathbb{E}\left[X_B^{(n_B)}\right]$, where κ is the maximum number of sellers that can be matched to *distinct* buyers that arrive after them[2] in σ: clearly, no mechanism can sell more than κ items. Bringing all together we have that

$$\mathcal{W}(\mathrm{OPT}) \leq n_s\mu_S + \kappa\mu_B^{(n_B)} = n_s\mu_S + r_B(n_B)\cdot\kappa\mu_B.$$

[1] As matter of fact, in the proof of Theorem 4 just regularity for the buyer values would suffice, i.e. F_B being MHR.

[2] You can think of that as the maximum size of a matching in the following undirected graph: the nodes are the sellers and the buyers, and there is an edge between any seller and all the buyers that appear after her in σ.

For the online algorithm now, from the sellers we get

$$\sum_{i \in N_S} \Pr[X_i > q] \, \mathbb{E}[X_i | X_i > q] \geq n_s (1 - F_S(q)) \, \mathbb{E}[X_S] = \frac{c_1 - 1}{c_1} \cdot n_S \mu_S$$

and from the buyers at least

$$\kappa \Pr[X_S \leq q] \Pr[X_B \geq p] \, \mathbb{E}[X_i | X_i \geq p] \geq \kappa F_S(q)(1 - F_B(p)) \, \mathbb{E}[X_B] = \frac{1}{c_1} \frac{1}{c_2} \cdot \kappa \mu_B,$$

just by considering one of the κ-size matchings discussed before: if we manage to buy from one of these κ sellers, then we will definitely have stock availability for the matched buyer. ◀

The upper bound in Lemma 5 cannot be improved:

▶ **Lemma 6.** *For* any *probability distribution F, even if the seller and buyer values are i.i.d. from F, the sequence SB^n forces all posted-price online mechanisms to have a competitive ratio of $\Omega(r(n))$, where $r(n) = \mu^{(n)}/\mu$ is the ratio of the n-maximum-order statistic of distribution F to its expectation.* ◀

As the following theorem demonstrates, our regularity assumption on the agent values is necessary if we want to hope for non-trivial bounds. In particular, the lower bound in Lemma 6 can be made arbitrarily high:

▶ **Theorem 7.** *For any constant $\varepsilon \in (0, 1)$, there exists a continuous probability distribution F such that any online posted-price mechanism has a competitive ratio of $\Omega(n^{1-\varepsilon})$ on the input sequence SB^n, even if the values of the sellers and the buyers are i.i.d.*

4.2 Profit

Now we turn our attention to our other objective of interest, that of maximizing the expected profit of the intermediary. As it turns out, this objective has some additional challenges that we need to address. For example, as the following theorem demonstrates, if the distribution of seller values is bounded away from 0, the competitive ratio can be arbitrarily bad, even for i.i.d. values from a uniform distribution. Intuitively, this follows from the impossibility of buying a super-constant number of items within a constant budget.

▶ **Theorem 8.** *For any $a > 0$ and $\varepsilon \in (0, 1)$, if the seller and buyer values are drawn i.i.d. from the uniform distribution over $[a, b]$ where $b > 2a$, then no online posted-price mechanism can have an approximation ratio better than $a \left(1 - \frac{1}{k}\right)^4 n^{1-\varepsilon}$ with respect to profit, where $k = \frac{b}{a} - 1$. In particular, for any uniform distribution over an interval $[1, h]$ with $h \geq 3$ the lower bound is $\frac{1}{2^4} n^{1-\varepsilon} = \Omega\left(n^{1-\varepsilon}\right)$.*

If we consider distributions supported over intervals that include 0, under our regularity assumptions we can do a little better than the trivial lower bound of Theorem 8:

▶ **Theorem 9.** *Under our regularity assumptions, for agent values distributed over intervals that include 0 the following online posted-price mechanism achieves a competitive ratio of $O(n^{\frac{1}{2}+\varepsilon})$ for any $\varepsilon > 0$:*

- *Post to the i-th seller price $q_i = F_S^{-1}\left(\frac{1}{e} \frac{1}{i^{1/2+\varepsilon}}\right)$*
- *Post to all buyers price $p = \mu_B$.*

Proof. Fix an input stream σ of length n. Let μ_B and s_B be the expectation and standard deviation of the buyer value distribution F_B. As in the proof of Lemma 5, let κ denote the maximum number of sellers that can be matched to distinct buyers that arrive after them in σ. If $\mu_B^{(j:m)}$ denotes the expectation of the j-th largest out of m independent draws from F_B, since no algorithm can make more than κ sales over its entire execution, the optimal offline profit is upper bounded by

$$\sum_{j=1}^{\kappa} \mu_B^{(n_B-j+1:n_B)} \leq \sum_{i=n-\kappa+1}^{n} \mu_B^{(i:n)} \leq \kappa\mu_B + 2\sqrt{\kappa n}s_B \leq 3\sqrt{\kappa}\sqrt{n}\mu_B,$$

where for the second inequality we have used Lemma 3 and for the last one we have used Property 4 from Theorem 1 and the obvious fact that $\kappa \leq n$.

For the analysis of the online mechanism now, the expected number of items that it gets from the first κ sellers is $\sum_{i=1}^{\kappa} F_S(q_i) = \frac{1}{e}\sum_{i=1}^{\kappa} \frac{1}{i^{1/2+\varepsilon}} \geq \frac{1}{e}\kappa^{1/2-\varepsilon}$. So, by considering the FIFO matching between these first κ sellers and their corresponding buyers, the expected income of our algorithm is at least $\frac{1}{e}\kappa^{1/2-\varepsilon}(1-F(p)) = \frac{1}{e}\kappa^{1/2-\varepsilon}(1-F(\mu_B)) \geq \frac{1}{e^2}\kappa^{1/2-\varepsilon}$, where in the last step we deployed Property 1 of Theorem 1. So, it only remains to be shown that the online algorithm does not spend more than a constant amount. Indeed, our expected spending is at most

$$\sum_{i=1}^{\infty} q_i F_S(q_i) \leq \sum_{i=1}^{\infty} e\mu_S F_S(q_i)^2 = \frac{1}{e}\mu_S \sum_{i=1}^{\infty} \frac{1}{i^{1+2\varepsilon}} = O(\mu_S),$$

where for the first inequality we have used Lemma 2, taking into consideration that seller prices q_i are decreasing and q_1 is below μ_S. This is true because again from Lemma 2 for $x = \mu_S$ we know that $\mu_S \leq e\mu_S F(\mu_S)$, or equivalently $F(\mu_S) \geq \frac{1}{e} = F(q_1)$. ◀

The algorithm of Theorem 9 is asymptotically optimal:

▶ **Theorem 10.** *If the seller and buyer values are drawn i.i.d. from the uniform distribution over $[0,1]$, then no online posted-price mechanism can have an approximation ratio better than $\Omega(\sqrt{n})$.*

Proof. We use the input sequence $\sigma = S^{n/2}B^{n/2}$ with n even. Let $F(x) = x$ be the cdf of the uniform distribution over $[0,1]$. This time we argue that no online algorithm can buy more than $\Omega(\sqrt{n})$ items from the sellers, in expectation. Indeed, let q_i be the price that the online mechanism posts to the i-th seller. Then, the expected number of items m_σ bought from the sellers is $\sum_{i=1}^{n/2} F(q_i) = \sum_{i=1}^{n/2} q_i$, while the expected expenditure c_σ is $\sum_{i=1}^{n/2} F(q_i)q_i = \sum_{i=1}^{n/2} q_i^2$. By the convexity of the function $t \mapsto t^2$ and Jensen's inequality it must be that

$$m_\sigma = \sum_{i=1}^{n/2} q_i \leq \sqrt{\frac{n}{2}} \left(\sum_{i=1}^{n/2} q_i^2\right)^{\frac{1}{2}} = O\left(\sqrt{c_\sigma}\sqrt{n}\right),$$

so given that our deficit must be $c_\sigma = O(\frac{1}{2})$, we get the desired $m_\sigma = O(\sqrt{n})$. As a result, the online profit can be at most $O(\sqrt{n}) \cdot 1 = O(\sqrt{n})$.

For the offline algorithm we use prices $q = \frac{1}{8}$ and $p = \frac{1}{2}$ for the buyers and sellers, respectively, and by an analogous analysis to that of the proof of Theorem 8, we get that the expected offline profit is at least

$$\frac{n}{2}F(q)(1-F(p))p - \frac{n}{2}F(q)q = \frac{n}{2}\frac{1}{8}\left(1-\frac{1}{2}\right)\frac{1}{2} - \frac{n}{2}\frac{1}{8}\frac{1}{8} = \frac{n}{128} = \Omega(n).$$ ◀

5 Limited Stock

If one looks carefully at the lower bound proof for the profit in Theorem 10, it becomes clear that the source of difficulty for any online algorithm is essentially the fact that without knowledge of the future, you cannot afford to spend a super-constant amount of money into accumulating a large stock of items, without the guarantee that there will be enough demand from future buyers. In particular, it may seem that the offline algorithm has an unrealistic advantage of using a stock of infinite size. The natural way to mitigate this would be to introduce an upper bound K on the number of items that both the online and offline algorithms can store at any point in time. As it turns out, this has a dramatic improvement in the competitive ratio for the profit:

▶ **Theorem 11.** *Assuming stock sizes of at most K items, under our regularity assumptions the following online mechanism is $O\left(Kr \log n\right)$-competitive, where $r = \max\left\{1, \frac{\mu_S}{\mu_B}\right\}$:*

- *If your stock is not currently full, post to sellers price $q = F_S^{-1}\left(\frac{1}{r}\frac{1}{2eK}\right)$*
- *Post to all buyers price $p = \mu_B$.*

Proof. The proof is similar to that of Theorem 9, but certain points need some special care. Let κ again be the maximum number of sellers that can be matched to distinct buyers that follow them, but this time under the added restriction of the K-size stock. This corresponds to the maximum matching with no "temporal" cut of size greater than K. We write "temporal" cut to mean any cut in the graph that separates the vertices (buyers and sellers) $1 \dots i$ from vertices $i+1 \dots n$ — that is, precisely the condition that we cannot match more than K sellers from an initial segment to buyers later in the sequence.

In the full version of our paper we show that such a κ-size matching can be computed not only offline, but also online using a FIFO queue of length K, adding sellers to the queue while it is not full and matching buyers greedily: we post prices to sellers, only if we have free space in our stock, i.e. when the matching queue is not full. We underestimate the online profit by considering only selling an item to the buyer that is matched to the seller from which we bought the item. Mimicking the analysis in the proof of Theorem 9 we can see that the expected number of items bought from the κ matched sellers is $\kappa F_S(q) \geq \kappa \frac{1}{2eK}\frac{1}{r}$.

Now we argue that $q \leq \frac{\mu_B}{2}$. Indeed, since $F_S(q) \leq \frac{1}{e}$ we know for sure that $q \leq \mu_S$, and so from Lemma 2 it is $q \leq e\mu_S F(q) \leq e\mu_S \frac{\mu_B}{\mu_S}\frac{1}{2e} = \frac{\mu_B}{2}$. Next, notice that whenever we make a successful sale, the contribution to profit is $p - q \geq \mu_B - \frac{\mu_B}{2} = \frac{1}{2}\mu_B$.

The rest of the proof can be found in the full version of the paper. ◀

▶ Remark. The above upper bound in Theorem 11, although a substantial improvement from the $\Theta(\sqrt{n})$ one for the general case in Theorem 9, cannot be improved further: the logarithmic lower bound is unavoidable, since a careful inspection of the welfare lower bound in the proof of Lemma 6 reveals that the same analysis carries over to the profit.

6 Balanced Sequences

As we saw in Section 5, introducing a restriction in the size of available stock can improve the performance of our online algorithms with respect to profit. However, the bound is still super-constant. Thus, it is perhaps more reasonable to assume some knowledge of the ratio α between buyers and sellers in sequences the intermediary might face. This allows us finer control over the trade-off between high volume of trades and the hunt for greater order statistics.

In this section we analyse the competitive ratio for profit and welfare obtained by online algorithms on α-*balanced* sequences.

▶ **Definition 12.** Let α be a positive integer. A sequence containing m buyers is called α-balanced if it contains αm sellers and the i-th buyer is preceded by at least αi sellers.

For example, the sequence $SBSSBSBB$ is 1-balanced, but $SBBSSB$ is not. Note that since $n = n_S \frac{\alpha+1}{\alpha} = n_B(\alpha+1)$, we only need to know the number of buyers of a sequence. For convenience, we will denote it by m instead of n_B, as it is used quite often. This constraint eliminates the pathological counterexamples of previous sections (such as SB^m) and introduces a much needed "recurrent" flavour to the market: items are constantly traded and in higher quantities, leading to greater profits for both online and offline algorithms.

6.1 Profit

We first work on profit, deriving bounds for a variety of online and offline mechanisms. Naturally, there are two types of offline mechanisms: adaptive and non-adaptive. The *non-adaptive* posted-price mechanism calculates all prices in advance based on the sequence of buyers and sellers, while the *adaptive* posted-price mechanism can alter the prices on the fly, depending on the outcomes of previous trades.

We show that there is a competitive online mechanism for α-balanced sequences. To do this, we compare the optimal adaptive and non-adaptive profit to the profit of a class of hypothetical mechanisms, called *fractional mechanisms*, which are allowed to buy fractional quantities of items: posting the price p would buy exactly $F_S(p)$ items or sell $1 - F_B(p)$ items. The advantage of using fractional mechanisms is that at any point we know the exact quantity of items in the hands of the intermediary instead of the expectation; an immediate consequence of this is that we know in advance whether there is enough quantity to sell, which implies that *the adaptive and non-adaptive versions of the optimal fractional mechanism are identical.*

We can now give an outline of the results in this section: For α-balanced sequences σ with m buyers and αm sellers, we establish the following relations of optimal profits:

$$\text{adaptive}(\sigma) \leq \text{fractional}(\sigma) \leq \text{fractional}(S^{\alpha m}B^m) \approx \text{non-adaptive}(\sigma), \tag{3}$$

the last of which will be our online algorithm. We begin by the fractional offline mechanism.

▶ **Theorem 13.** *The profit gained by the optimal fractional mechanism for the sequence $S^{\alpha m}B^m$ is*

$$\begin{aligned} \max \quad & m\left(p(1-F_B(p)) - \alpha \cdot qF_S(q)\right) \\ s.t. \quad & 1 - F_B(p) = \alpha F_S(q) \\ & p, q \in [0, \infty). \end{aligned} \tag{4}$$

For other sequences containing αm sellers and m buyers in a different order, we can use the following lemma to establish the middle part of inequality 3.

▶ **Lemma 14.** *For any α-balanced σ with m buyers, fractional$(\sigma) \leq$ fractional$(S^{\alpha m}B^m)$*

▶ **Theorem 15.** *For any sequence σ we have adaptive$(\sigma) \leq$ fractional(σ).*

The intuition behind the proof of the theorem is that the optimal adaptive profit is bounded from above by the optimal fractional adaptive profit (since fractional mechanisms is a more

general class of mechanisms); since in fractional mechanisms optimal adaptive and non-adaptive profits are the same, the theorem follows. For a more rigorous technical treatment, see the full version of our paper.

At this point, we have a clear model of the adversary's power: the fractional mechanism's revenue for sequence $S^{\alpha m}B^m$, setting only two prices p, q for sellers and buyers. Could we do the same online? It seems likely. After all, long sequences of buyers and sellers seem to lead to a similar amount of trading on average by a mechanism setting the same prices.

Based on the previous discussion we propose the following online posted price algorithm:

- Use prices p, q given by the optimal fractional solution for $S^{\alpha m}B^m$(see Theorem 13).

This algorithm works without knowing the length of the sequence chosen by the adversary.

▶ **Lemma 16.** *Let A be the online algorithm defined by the optimal fractional offline prices of (4). Consider two α-balanced sequences σ_1 and σ_2 of equal length. We write $\sigma_1 \succ \sigma_2$ whenever every prefix of σ_1 contains more sellers than the prefix of σ_2 having equal length. Then, $\sigma_1 \succ \sigma_2 \Rightarrow \mathcal{R}(A, \sigma_1) \geq \mathcal{R}(A, \sigma_2)$*

Although not all sequences are comparable (e.g. $SSBBSB$ and $SBSSBB$), the sequence $(S^\alpha B)^m$ is the bottom element among all α-balanced sequences of length $(\alpha+1)m$. This is trivial, as any balanced sequence must have at least $\lceil \frac{i}{(\alpha+1)/(\alpha)} \rceil$ sellers for any prefix of length i and $(S^\alpha B)^m$ is tight for this bound.

To formalize our intuition of making the same number of trades in the long run, we reformulate our algorithm in the more familiar setting of random walks. Instead of considering agents separately, each "timestep" would be one sub-sequence $S^\alpha B$, giving m steps in total. Thus, we are interested in the random variables Z_i, denoting the items in stock at the end of each step, starting with $Z_0 = 0$. Knowing the algorithm buys $\alpha m F_S(q)$ items in expectation, the expected profit can be given by

$$\mathcal{R}((S^\alpha B)^m) = (\alpha m F_S(q) - \mathbb{E}[Z_m])(p-q) - \mathbb{E}[Z_m]\, q, \tag{5}$$

which is the revenue of the expected number of trades minus the cost of the unsold items.

▶ **Lemma 17.** $\mathbb{E}[Z_m] \leq \sqrt{2m\alpha^2 \log m}\left(1 - \frac{2}{m}\right) + 2$

Proof. The process Z_i is almost a martingale but not quite: clearly $\mathbb{E}[Z_i] \leq \alpha m$ for all i and we do have $\mathbb{E}[Z_{i+1}|Z_i \geq 1] = Z_i$ since the expected change in items after that step is $\alpha F_S(q) - (1 - F_B(p)) = 0$ by Theorem 13 . However, $\mathbb{E}[Z_{i+1}|Z_i = 0] > Z_i$, by the no short selling assumption.

We can define Y_i in the same probability space, where $Y_0 = 0$, and

$$Y_{i+1} = Y_i + \begin{cases} Z_{i+1} & \text{if } Y_i > 0 \\ -Z_{i+1} & \text{if } Y_i < 0 \\ \begin{cases} Z_{i+1} & \text{with probability } \frac{1}{2} \\ -Z_{i+1} & \text{with probability } \frac{1}{2} \end{cases} & \text{if } Y_i = 0 \end{cases}. \tag{6}$$

The crucial observation is that Y_i behaves similar to Z_i but has no barrier at 0. Notice, that $|Y_i| \geq Z_i$ for all i and Y_i is a martingale.

Moreover, we have that $|Y_{i+1} - Y_i| \leq \alpha$ thus by the Azuma-Hoeffding inequality we can bound the expected value $\mathbb{E}[Z_m]$:

$$\Pr[Z_m \geq x] \leq \Pr[|Y_m| \geq x] = \Pr[|Y_m - Y_0| \geq x] \leq 2e^{\frac{-x^2}{2m\alpha^2}} \Rightarrow \tag{7}$$

$$\mathbb{E}[Z_m] \leq x\left(1 - 2e^{\frac{-x^2}{2m\alpha^2}}\right) + 2\alpha m e^{\frac{-x^2}{2m\alpha^2}}, \tag{8}$$

where we can set $x = \sqrt{2m\alpha^2 \log m}$ to obtain the simpler form:

$$\mathbb{E}[Z_m] \leq \sqrt{2m\alpha^2 \log m}\left(1 - \frac{2}{m}\right) + 2\alpha. \tag{9}$$

◀

▶ **Lemma 18.** *Let $r = \max\left\{2, \frac{\mu_S}{\mu_B}\right\}$. The optimal value of Programme (4) is at least $m\frac{\mu_B}{2er}$. Furthermore, at any optimal solution the buyer price has to be at most $p \leq 4\ln(4er)\mu_B$.*

▶ **Theorem 19.** *Under our regularity assumptions, the proposed non-adaptive online mechanism is $(1 + o(\alpha^{3/2} r \log r))$-competitive for any balanced sequence, where $r = \max\left\{2, \frac{\mu_S}{\mu_B}\right\}$.*

Proof. Plugging (9) into (5), we get:

$$\begin{aligned}
\mathcal{R}((S^\alpha B)^m) &\geq \alpha m F_S(q)(p-q) - \mathbb{E}[Z_m](p-q) - \mathbb{E}[Z_m] q \\
&\geq \alpha m F_S(q)(p-q) - \left(\sqrt{2m\alpha^2 \log m}\left(1 - \frac{2}{m}\right) + 2\alpha\right) p \\
&\geq \alpha m F_S(q)(p-q) - O(\alpha\sqrt{m \ln mp}).
\end{aligned} \tag{10}$$

Using Lemma 14, Theorem 15 and Theorem 13 we know that for every α-balanced sequence, the profit of our non-adaptive online algorithm is at least $\mathcal{R}((S^\alpha B)^m)$ and the optimal offline is at most that of the fractional on sequence $S^{\alpha m} B^m$, i.e. $\alpha m F_S(q)(p-q)$. Thus, the second term in (10) bounds the additive difference of the online and optimal offline profit, and its ratio with respect to the offline profit is upper bounded by

$$O\left(\frac{\alpha\sqrt{m \ln mp}}{\alpha m F_S(q)(p-q)}\right) = O\left(\frac{\alpha\sqrt{m \ln m}\mu_B \ln(4er)}{m\frac{\mu_B}{2er}}\right) = O\left(\alpha^{3/2}\sqrt{\frac{\ln n}{n}} r \log r\right). \qquad ◀$$

▶ Remark. Among all 1-balanced sequences, the sequence that gives the maximum profit is not $S^m B^m$; intuitively, by moving buyers earlier in the sequence, we obtain more profit by adapting the remaining buying prices to the outcome of these potential trades. For example, the sequence $S^{m/2} B S^{m/2} B^{m-1}$ has better adaptive profit than the sequence $S^m B^m$ for large m. Our work above shows that the difference is asymptotically insignificant, but it remains an intriguing question to determine the balanced sequence with the maximum profit.

6.2 Welfare

Welfare on balanced sequences also improves the competitive ratio of Theorem 4 to a constant. Intuitively, the reason is that the high volume of possible trades dampens the advantage the adversary has in obtaining higher order statistics from buyers.

▶ **Theorem 20.** *The online auction that posts to any seller and buyer the median of their distribution is 4-competitive.*

Notice the above theorem holds without any regularity assumption on the agent value distributions.

Acknowledgements. We want to thank Matthias Gerstgrasser for many helpful discussions and his assistance during the initial development of our paper.

References

1 Moshe Babaioff, Liad Blumrosen, Shaddin Dughmi, and Yaron Singer. Posting Prices with Unknown Distributions. In *Innovations in Computer Science (ICS)*, jan 2011. URL: http://conference.itcs.tsinghua.edu.cn/ICS2011/content/paper/35.pdf.

2 Moshe Babaioff, Shaddin Dughmi, Robert D. Kleinberg, and Aleksandrs Slivkins. Dynamic pricing with limited supply. *ACM Trans. Economics and Comput.*, 3(1):4, 2015. doi:10.1145/2559152.

3 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7(2):7:1–7:11, June 2008. doi:10.1145/1399589.1399596.

4 Ashwinkumar Badanidiyuru, Robert Kleinberg, and Yaron Singer. Learning on a budget: posted price mechanisms for online procurement. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 128–145. ACM, 2012. URL: http://dl.acm.org/citation.cfm?id=2229026.

5 Ziv Bar-Yossef, Kirsten Hildrum, and Felix Wu. Incentive-compatible online auctions for digital goods. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'02, pages 964–970, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=545381.545506.

6 Richard E. Barlow and Albert W. Marshall. Bounds for Distributions with Monotone Hazard Rate, I. *The Annals of Mathematical Statistics*, 35(3):1234–1257, sep 1964. URL: http://projecteuclid.org/euclid.aoms/1177703281, doi:10.1214/aoms/1177703281.

7 Avrim Blum and Jason D. Hartline. Near-optimal online auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1156–1163. Society for Industrial and Applied Mathematics, 2005. URL: http://dl.acm.org/citation.cfm?id=1070597.

8 Avrim Blum, Tuomas Sandholm, and Martin Zinkevich. Online algorithms for market clearing. *Journal of the ACM (JACM)*, 53(5):845–879, 2006. URL: http://dl.acm.org/citation.cfm?id=1183913.

9 Liad Blumrosen and Shahar Dobzinski. (Almost) Efficient Mechanisms for Bilateral Trading. *arXiv preprint arXiv:1604.04876*, 2016. URL: http://arxiv.org/abs/1604.04876.

10 Liad Blumrosen and Thomas Holenstein. Posted prices vs. negotiations: An asymptotic analysis. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, EC'08, pages 49–49, New York, NY, USA, 2008. ACM. doi:10.1145/1386790.1386801.

11 Liad Blumrosen and Yehonatan Mizrahi. Approximating gains-from-trade in bilateral trading. In *Web and Internet Economics*, pages 400–413. Springer, Berlin, Heidelberg, December 2016. doi:10.1007/978-3-662-54110-4_28.

12 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

13 Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 311–320. ACM, 2010. URL: http://dl.acm.org/citation.cfm?id=1806733.

14 Riccardo Colini-Baldeschi, Bart de Keijzer, Stefano Leonardi, and Stefano Turchetta. Approximately efficient double auctions with strong budget balance. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1424–1443. Society for Industrial and Applied Mathematics, 2016.

15 X Deng, P Goldberg, B Tang, and J Zhang. Revenue maximization in a bayesian double auction market. *Theoretical Computer Science*, 2014. doi:10.1016/j.tcs.2014.04.013.

16 Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 123–135. Society for Industrial and Applied Mathematics, 2015.

17 Matthias Gerstgrasser, Paul W Goldberg, and Elias Koutsoupias. Revenue maximization for market intermediation with correlated priors. In *International Symposium on Algorithmic Game Theory*, pages 273–285. Springer, 2016.

18 Yiannis Giannakopoulos, Elias Koutsoupias, and Philip Lazos. Online market intermediation. *CoRR*, abs/1703.09279, 2017. URL: http://arxiv.org/abs/1703.09279.

19 Yiannis Giannakopoulos and Maria Kyropoulou. The VCG Mechanism for Bayesian Scheduling. In *Proceedings of the 11th Conference on Web and Internet Economics*, volume 9470 of *Lecture Notes in Computer Science*, pages 343–356. Springer, 2015. URL: http://arxiv.org/abs/1509.07455, doi:10.1007/978-3-662-48995-6_25.

20 Mohammad T. Hajiaghayi. Online auctions with re-usable goods. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 165–174. ACM, 2005. URL: http://dl.acm.org/citation.cfm?id=1064027.

21 Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 71–80. ACM, 2004. URL: http://dl.acm.org/citation.cfm?id=988784.

22 Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, volume 7, pages 58–65, 2007.

23 Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'03, pages 594–, Washington, DC, USA, 2003. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=946243.946352.

24 Elias Koutsoupias and George Pierrakos. On the competitive ratio of online sampling auctions. *ACM Transactions on Economics and Computation*, 1(2):10, May 2013. URL: http://dl.acm.org/citation.cfm?id=2465769.2465775.

25 R. Preston McAfee. A dominant strategy double auction. *Journal of Economic Theory*, 56(2):434–450, 1992. doi:10.1016/0022-0531(92)90091-U.

26 Roger B Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983.

27 Rad Niazadeh, Yang Yuan, and Robert Kleinberg. Simple and near-optimal mechanisms for market intermediation. In *International Conference on Web and Internet Economics*, pages 386–399. Springer, 2014.

28 David C. Parkes. Online mechanisms. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, chapter 16. Cambridge University Press, New York, NY, USA, 2007.

29 Qiqi Yan. Mechanism design via correlation gap. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'11, pages 710–719. SIAM, 2011. URL: http://dl.acm.org/citation.cfm?id=2133036.2133092.

Tight Lower Bounds for Multiplicative Weights Algorithmic Families*

Nick Gravin[1], Yuval Peres[2], and Balasubramanian Sivan[3]

1 **Massachusetts Institute of Technology, Cambridge, MA, USA**
ngravin@mit.edu
2 **Microsoft Research, Redmond, WA, USA**
peres@microsoft.com
3 **Google Research, New York, NY, USA**
balusivan@google.com

Abstract

We study the fundamental problem of prediction with expert advice and develop regret lower bounds for a large family of algorithms for this problem. We develop simple adversarial primitives, that lend themselves to various combinations leading to sharp lower bounds for many algorithmic families. We use these primitives to show that the classic Multiplicative Weights Algorithm (MWA) has a regret of $\sqrt{\frac{T\ln k}{2}}$ (where T is the time horizon and k is the number of experts), there by completely closing the gap between upper and lower bounds. We further show a regret lower bound of $\frac{2}{3}\sqrt{\frac{T\ln k}{2}}$ for a much more general family of algorithms than MWA, where the learning rate can be arbitrarily varied over time, or even picked from arbitrary distributions over time. We also use our primitives to construct adversaries in the geometric horizon setting for MWA to precisely characterize the regret at $\frac{0.391}{\sqrt{\delta}}$ for the case of 2 experts and a lower bound of $\frac{1}{2}\sqrt{\frac{\ln k}{2\delta}}$ for the case of arbitrary number of experts k (here δ is the probability that the game ends in any given round).

1998 ACM Subject Classification F.2.0 [Analysis of Algorithms and Problem Complexity] General

Keywords and phrases Multiplicative Weights, Lower Bounds, Adversarial Primitives

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.48

1 Introduction

In this paper we develop tight lower bounds on the regret obtainable by a broad family of algorithms for the fundamental problem of prediction with expert advice. Predicting future events based on past observations, a.k.a. prediction with expert advice, is a classic problem in learning. The experts framework was the first framework proposed for online learning and encompasses several applications as special cases. The underlying problem is an online optimization problem: a *player* has to make a decision at each time step, namely, decide which of the k experts' advice to follow. At every time t, an *adversary* sets gains for each expert: a gain of $g_{i,t}$ for expert i at time t. Simultaneously, the *player*, seeing the gains from all previous steps except t, has to choose an action, i.e., decide on which expert to follow. If the player follows expert $j(t)$ at time t, he gains $g_{j(t),t}$. At the end of each step t, the gains

* A full version of the paper is available at http://arxiv.org/abs/1607.02834.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 48; pp. 48:1–48:14

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

associated with all experts are revealed to the player, and the player's choice is revealed to the adversary. In the *finite horizon model*, this process is repeated for T steps, and the player's goal is to perform (achieve a cumulative gain) as close as possible to the best single action (best expert) in hindsight, i.e., to minimize his *regret* $R_{T,k}$:

$$R_{T,k} = \max_{1 \le i \le k} \sum_{t=1}^{T} g_{i,t} - \sum_{t=1}^{T} g_{j(t),t}.$$

Apart from assuming that the $g_{i,t}$'s are bounded in $[0,1]$, we don't assume anything else about the gains[1]. Just as natural as the finite horizon model is the model with a *geometric horizon*: the stopping time is a geometric random variable with expectation $\frac{1}{\delta}$. In other words, the process ends at any given step with probability δ, independently of the past. Equivalently, both the player and the adversary discount the future with a $1-\delta$ factor. In this paper, we study both the finite horizon model and the geometric horizon model. We begin with the discussion for finite horizon model below.

Main contribution

In this paper we develop simple adversarial primitives and demonstrate that, when applied in various combinations, they result in remarkably sharp lower bounds for a broad family of algorithms. We first describe the family of algorithms we study, and then discuss our main results.

Multiplicative Weights Algorithm

We begin with the Multiplicative Weights Algorithm, which is a simple, powerful and widely used algorithm for a variety of learning problems. In the experts problem, at each time t, MWA computes the cumulative gain $G_{i,t-1} = \sum_{s=1}^{t-1} g_{i,s}$ of each expert i accumulated over the past $t-1$ steps, and will follow expert i's advice with probability proportional to $e^{\eta G_{i,t-1}}$. Namely, with probability $\frac{e^{\eta G_{i,t-1}}}{\sum_{j=1}^{k} e^{\eta G_{j,t-1}}}$ where η is a parameter that can be tuned. The per-step computation of the algorithm is extremely simple and straightforward. The intuition behind the algorithm is to increase the weight of any expert that performs well by a multiplicative factor. Despite the simplicity and the heuristic origins of the algorithm, it is surprisingly powerful: the pioneering work of Cesa Bianchi et al. [6] showed that MWA obtains a sublinear regret of $\sqrt{\frac{T \ln k}{2}}$, and that this is asymptotically optimal as the number of experts k and the number of time steps T both tend to ∞.

Families of algorithms

The MWA is a single-parameter family of algorithms, i.e., the learning rate parameter η is the only parameter available for the player. In general one could think of η being an arbitrary function of time t, i.e., at step t, algorithm follows expert i with probability $\frac{e^{\eta(t) G_{i,t-1}}}{\sum_{j=1}^{k} e^{\eta(t) G_{j,t-1}}}$. Note that this is a T-parameter family of algorithms and is quite general. The most general family of algorithms we study is when at each time t, the quantity $\eta(t)$ is drawn from an

[1] As one might expect, it turns out that restricting the adversary to set gains in $\{0,1\}$ instead of $[0,1]$ is without loss of generality (see [12] or [20]). Henceforth, we restrict ourselves to the binary adversary, which just sets gains of 0 or 1.

arbitrary distribution F_t over reals. Since F_t could be arbitrary, this is an infinite-parameter family of algorithms. We denote the

1. single parameter MWA family by $\mathcal{A}_{\text{single}}$;
2. family where $\eta(t)$ decreases with t by $\mathcal{A}_{\text{dec}}$;
3. family where $\eta(t)$ is arbitrary function of t by $\mathcal{A}_{\text{arb}}$;
4. family where $\eta(t)$ is drawn from F_t for each t by $\mathcal{A}_{\text{rand}}$.

It is straightforward to see that $\mathcal{A}_{\text{single}} \subseteq \mathcal{A}_{\text{dec}} \subseteq \mathcal{A}_{\text{arb}} \subseteq \mathcal{A}_{\text{rand}}$. The reason we start with $\mathcal{A}_{\text{single}}$ is that it is the classic MWA and precisely characterizing its regret is still open. We study $\mathcal{A}_{\text{dec}}$ because often when MWA algorithms are working with unknown T, they employ a strategy where η decreases with time. We move on to further significantly generalize this by studying $\mathcal{A}_{\text{arb}}, \mathcal{A}_{\text{rand}}$.

Minimax regret, and Notation

We study the standard notion of minimax regret for each of the above family of algorithms. Formally, let $R_{T,k}(A, D)$ denote the expected regret achieved by algorithm A when faced with adversary D in the prediction with expert advice game with T steps and k experts. We use $R_k(A, D)$ to denote the asymptotic[2], in T, value of $R_{T,k}(A, D)$, i.e., $R_k(A, D) = \sqrt{T} \cdot \lim_{T\to\infty} \frac{R_{T,k}(A,D)}{\sqrt{T}}$. The minimax regret of a family $\mathcal{A}_{\mathcal{F}}$ of algorithms against a family $\mathcal{D}_{\mathcal{F}}$ of adversaries is given by $R_{T,k}(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{\mathcal{F}}) = \min_{A\in\mathcal{A}_{\mathcal{F}}} \max_{D(A)\in\mathcal{D}_{\mathcal{F}}} R_{T,k}(A, D)$. Let $\mathcal{D}_{\text{univ}}$ denote the universe of all adversaries. We use the shorthand $R_{T,k}(\mathcal{A}_{\mathcal{F}})$ for $R_{T,k}(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{\text{univ}}) = \min_{A\in\mathcal{A}_{\mathcal{F}}} \max_{D(A)\in\mathcal{D}_{\text{univ}}} R_{T,k}(A, D)$. We use $R_k(\mathcal{A}_{\mathcal{F}})$ to denote the asymptotic, in T, value of $R_{T,k}(\mathcal{A}_{\mathcal{F}})$, i.e., $R_k(\mathcal{A}_{\mathcal{F}}) = \sqrt{T} \cdot \lim_{T\to\infty} \frac{R_{T,k}(\mathcal{A}_{\mathcal{F}})}{\sqrt{T}}$.

Goal

One of our goals in this paper is to compute the precise values of $R_k(\mathcal{A}_{\text{single}})$, $R_k(\mathcal{A}_{\text{dec}})$, $R_k(\mathcal{A}_{\text{arb}})$ and $R_k(\mathcal{A}_{\text{rand}})$ for each value of k, and, *describe and compute* the adversarial sequences that realize these regrets. For clarity, we compute the precise values of $R_k(\mathcal{A}_{\mathcal{F}})$ by:

1. computing the best-response adversary in $\mathcal{D}_{\text{univ}}$ for every algorithm in $\mathcal{A}_{\mathcal{F}}$;
2. computing $R_k(\mathcal{A}_{\mathcal{F}})$ the regret of the optimal algorithm in $\mathcal{A}_{\mathcal{F}}$ (i.e., the algorithm that gets the smallest regret w.r.t. its best-response adversary).

In many cases, the first step, namely computing the best-response adversary, is challenging. We find the best-response adversaries for the families $\mathcal{A}_{\text{single}}$ and $\mathcal{A}_{\text{dec}}$. For the families $\mathcal{A}_{\text{arb}}$ and $\mathcal{A}_{\text{rand}}$, we perform the first step approximately, i.e., we compute a nearly best-response adversary, and thus we obtain lower bounds on $R_k(\mathcal{A}_{\text{arb}})$ and $R_k(\mathcal{A}_{\text{rand}})$.

What is known, and what to expect?

It is well known that for $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{\text{single}}, \mathcal{A}_{\text{dec}}, \mathcal{A}_{\text{arb}}, \mathcal{A}_{\text{rand}}$: $R_{T,k}(\mathcal{A}_{\mathcal{F}}) \le \sqrt{(T \ln k)/2}$ for all T, k, and in the doubly asymptotic limit, as both T and k go to ∞, the optimal regret of $\mathcal{A}_{\text{single}}$ is $\sqrt{(T \ln k)/2}$, i.e., $\lim_{T\to\infty,k\to\infty} \left(R_{T,k}(\mathcal{A}_{\text{single}})/\sqrt{(T \ln k)/2}\right) = 1$. (see [6, 5]). While there are useful applications for $k \to \infty$, there are also several interesting use-cases of the experts problem with just a few experts (rain-or-shine ($k = 2$), buy-or-sell-or-hold ($k = 3$)). It seems like for small k such as $2, 3, 4$ etc. $R_k(\mathcal{A}_{\text{single}})$ could be a significant constant factor smaller than $\sqrt{(T \ln k)/2}$. And given that families like $\mathcal{A}_{\text{dec}}$ etc. are supersets of $\mathcal{A}_{\text{single}}$,

[2] Although R_k doesn't have a T in the subscript, R_k is still dependent on T. We suppress T merely to indicate asymptotics in T.

it seems even more likely that $R_k(\mathcal{A}_{\text{dec}})$ etc. are constant factor smaller than $\sqrt{(T \ln k)/2}$. Surprisingly, we show that is not the case: the regret of $\sqrt{(T \ln k)/2}$ that is obtained as $k \to \infty$ is *already obtained at* $k = 2$. Thus our work completely closes the gap between upper and lower bounds for all k.

1.1 Main Results

Finite horizon model

1. $R_k(\mathcal{A}_{\text{single}}) = R_k(\mathcal{A}_{\text{dec}}) = \sqrt{\frac{T \ln k}{2}}$ for even k,
   $R_k(\mathcal{A}_{\text{single}}) \geq R_k(\mathcal{A}_{\text{dec}}) \geq \sqrt{\frac{T \ln k}{2}(1 - \frac{1}{k^2})}$ for odd k.
2. $R_k(\mathcal{A}_{\text{arb}}) \geq R_k(\mathcal{A}_{\text{rand}}) \geq \frac{2}{3}\sqrt{\frac{T \ln k}{2}}$ for even k,
   $R_k(\mathcal{A}_{\text{arb}}) \geq R_k(\mathcal{A}_{\text{rand}}) \geq \frac{2}{3}\sqrt{\frac{T \ln k}{2}(1 - \frac{1}{k^2})}$ for odd k.

Geometric horizon model

In the geometric horizon model, the current time t is not relevant, since the expected remaining time for which the game lasts is the same irrespective of how many steps have passed in the past. Thus $\eta(t)$ is without loss of generality, independent of t. Nevertheless, η could still depend on other aspects of the history of the game, like the cumulative gains of all the experts etc. We establish some quick notation before discussing results. Let δ denote the probability that the game stops at any given step, independently of the past (and therefore the expected length of the game is $\frac{1}{\delta}$). Let $R_{\delta,k}(A, D)$ denote the regret achieved by algorithm A when faced with adversary D in the prediction with expert advice game with stopping probability δ and k experts. The minimax regret for a family $\mathcal{A}_{\mathcal{F}}$ of algorithms is given by $R_{\delta,k}(\mathcal{A}_{\mathcal{F}}) = R_{\delta,k}(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{\text{univ}}) = \min_{A \in \mathcal{A}_{\mathcal{F}}} \max_{D \in \mathcal{D}_{\text{univ}}} R_{\delta,k}(A, D)$. Let[3] $R_k(\mathcal{A}_{\mathcal{F}}) = \frac{1}{\sqrt{\delta}} \lim_{\delta \to 0} \sqrt{\delta} \cdot R_{\delta,k}(\mathcal{A}_{\mathcal{F}})$.

We show the following:

1. $R_2(\mathcal{A}_{\text{single}}) = \frac{0.391}{\sqrt{\delta}}$,
2. $R_k(\mathcal{A}_{\text{single}}) \geq \frac{1}{2}\sqrt{\frac{\ln k}{2\delta}}$ for all k.

The regret lower bound of $\frac{1}{2}\sqrt{\frac{\ln k}{2\delta}}$ we obtain is at most a factor 2 away from the regret upper bound of $\sqrt{\frac{\ln k}{2\delta}}$. Further, we show that the adversarial family that we use for the family of algorithms $\mathcal{A}_{\text{single}}$ to obtain the precise regret for 2 experts, also obtains the optimal regret for the universe $\mathcal{A}_{\text{univ}}$ of *all algorithms*. See See the full version [11] for more on this result.

1.2 Simple adversarial primitives and families

While the optimal regret $R_k(\mathcal{A}_{\mathcal{F}})$ is defined by optimizing over the most general family $\mathcal{D}_{\text{univ}}$ of all adversaries, (i.e., $R_k(\mathcal{A}_{\mathcal{F}}) = R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{\text{univ}})$) one of our primary contributions in this work is to develop simple and analytically easy-to-work-with adversarial primitives that we use to construct adversarial families (call a typical such family $\mathcal{D}_{\text{simple}}$) such that:

[3] Note that the notation $R_k(\cdot)$ is overloaded: it could refer to finite or geometric horizon setting depending on the context. But since the setting is clear from the context, we drop the δ vs T.

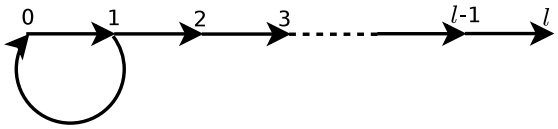


Figure 1 Optimal finite horizon adversary.

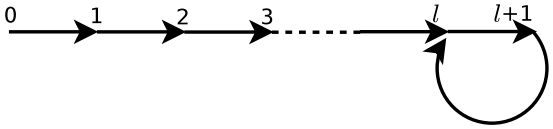


Figure 2 Optimal geometric horizon adversary.

- $\mathcal{D}_{\text{simple}}$ is simple to-describe and to-optimize-over, i.e., computing $\max_{D\in\mathcal{D}_{\text{simple}}} R_{T,k}(A,D)$ is much simpler than computing $\max_{D\in\mathcal{D}_{\text{univ}}} R_{T,k}(A,D)$.
- optimizing over $\mathcal{D}_{\text{simple}}$ is guaranteed to be as good (or approximately as good) as optimizing over $\mathcal{D}_{\text{univ}}$ for many algorithmic families $\mathcal{A}_{\mathcal{F}}$, i.e., $R_k(\mathcal{A}_{\mathcal{F}},\mathcal{D}_{\text{univ}}) = R_k(\mathcal{A}_{\mathcal{F}},\mathcal{D}_{\text{simple}})$ for many $\mathcal{A}_{\mathcal{F}}$. As $R_k(\mathcal{A}_{\mathcal{F}},\mathcal{D}_{\text{univ}}) \geq R_k(\mathcal{A}_{\mathcal{F}},\mathcal{D}_{\text{simple}})$, the non-trivial part is to prove (approximate) equality for $\mathcal{A}_{\mathcal{F}}$.

We demonstrate the versatility of our primitives by using simple combinations of them to develop sharp lower bounds to algorithmic families $\mathcal{A}_{\text{single}}$, $\mathcal{A}_{\text{dec}}$, $\mathcal{A}_{\text{arb}}$, and $\mathcal{A}_{\text{rand}}$. There is a lot of room for further combinations of primitives that might be useful to construct adversarial families tailored to other algorithmic families.

The "looping" and "straight-line" primitives

These primitives are best described by focusing on the case of $k = 2$ experts. In the two experts case, the algorithm makes its decision at step t, by just looking at the difference d of the cumulative gains of the leading and lagging experts' cumulative gains. As such, the adversary has to simply control how the difference d evolves over time. The "looping" primitive simply loops the value of d between 0 and 1 indefinitely (i.e., advances[4] one expert in one step and advances the other in the next step and so on, so that d simply loops between 0 and 1). The "straight-line" adversary simply keeps advancing the value of d by 1 at each step. Interestingly, the worst-case adversary for each of the finite and geometric horizon settings is a composition of looping and straight-line primitives. Strikingly, despite the apparent similarity between two settings, the optimal adversaries in the two models turn out to be "mirror images" of each other. The optimal adversary in finite horizon loops first and then goes in straight-line, while the geometric horizon's optimal does the reverse. The structure of these two families is depicted in Figures 1, 2, that shows the evolution of the difference d between the cumulative gains of the leading and lagging experts. This fundamental difference between the structures of the optimal adversary in these two settings also manifests in the optimal regret values of these two settings.

The generalizations of these primitives for arbitrary k is straightforward. The looping primitive partitions the set of experts into two teams, say A and B, and then it advances all experts in team A in one step and in team B in the other, and so on. The straight-line primitive picks an arbitrary expert and keeps advancing that expert by 1 in each step.

[4] Advances here refers to setting the gain of that expert to 1.

Combining the primitives

Here's how we create effective adversarial families from these primitives. In fact the families are often trivial, i.e., they have only one member and therefore there's nothing to optimize. We ignore the odd and even k distinctions here for ease of description and just focus on the even k case. Please see the technical sections for precise descriptions, which is only slightly different from what is here.

1. Perform $\frac{T-\ell}{2}$ loops and then ℓ straight-line steps, for $\ell = T^{3/4}$. Call this adversary D_{lsdet} (stands for loop-straight-deterministic). Clearly, this adversarial family is simple-to-describe and there is nothing to optimize here as there is only one member in the family. Most importantly, it gives the precisely optimal regret for algorithmic families $\mathcal{A}_{\text{single}}$ and $\mathcal{A}_{\text{dec}}$ as $T \to \infty$. I.e.,

$$\text{For families } \mathcal{A}_{\mathcal{F}} = \mathcal{A}_{\text{single}}, \mathcal{A}_{\text{dec}}: \qquad R_k(\mathcal{A}_{\mathcal{F}}, D_{lsdet}) = \sqrt{\frac{T \ln k}{2}} = R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{\text{univ}}).$$

   The best known regret lower bound for $\mathcal{A}_{\text{single}}$ was $\frac{1}{4}\sqrt{T \log_2 k}$ [13], which leaves a factor 2.35 gap between upper and lower bounds, that our work closes. We are not aware of prior lower bounds for $\mathcal{A}_{\text{dec}}$.
2. Perform $\frac{T-r}{2}$ loops and then r straight-line steps, where r is chosen uniformly at random from $\{0, 1, \ldots, T^{3/4}\}$. This family is simple and there is nothing to optimize here as well. Call this adversary D_{lsrand} (denoting loop, straight, uniformly random). We show that when $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{\text{arb}}$ or when $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{\text{rand}}$:

$$\text{For families } \mathcal{A}_{\mathcal{F}} = \mathcal{A}_{\text{arb}}, \mathcal{A}_{\text{rand}}: \qquad R_k(\mathcal{A}_{\mathcal{F}}, D_{lsrand}) \geq \frac{2}{3}\sqrt{\frac{T \ln k}{2}} \geq \frac{2}{3} R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{\text{univ}}).$$

   Note that while this lower bound doesn't precisely match the upper bound, the upper bound $R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{\text{univ}}) \leq \sqrt{(T \ln k)/2}$ and is likely even smaller for small k (particularly for a large family of algorithms like $\mathcal{A}_{\text{arb}}$ or $\mathcal{A}_{\text{rand}}$) — *thus our result shows that the ratio between upper and lower bounds is at most* $\frac{3}{2}$ and likely even smaller. To the best of our knowledge our lower bound is the first for the classes $\mathcal{A}_{\text{arb}}$ and $\mathcal{A}_{\text{rand}}$.
3. In geometric horizon, even for the family $\mathcal{A}_{\text{single}}$ and at $k = 2$ experts, instead of a single adversary working for all members of $\mathcal{A}_{\text{single}}$, we have a single-parameter family of adversaries to optimize over. Namely, follow the straight-line primitive for r steps and then the looping primitive for $\frac{T-r}{2}$ steps. Call this single-parameter family (parameterized by r) as $\mathcal{D}_{sl}$. The exact number r is determined by optimizing it as a function of the parameter η used by the algorithm in $\mathcal{A}_{\text{single}}$. Specifically, for the case of 2 experts we show that: $R_2(\mathcal{A}_{\text{single}}, \mathcal{D}_{sl}) = \frac{0.391}{\sqrt{\delta}} = R_2(\mathcal{A}_{\text{single}}, \mathcal{D}_{\text{univ}})$. Note that D_{sl} is again simple-to-describe and straightforward-to-optimize over. Further, it is the precisely optimal adversary family for not just $\mathcal{A}_{\text{single}}$ but also the *universe of all algorithms* $\mathcal{A}_{\text{univ}}$ (see the full version [11] for this result), i.e., $R_2(\mathcal{A}_{\text{univ}}, \mathcal{D}_{\text{univ}}) = R_2(\mathcal{A}_{\text{univ}}, \mathcal{D}_{sl})$.
4. But in the geometric horizon setting, if we don't shoot for the precisely optimal adversary family, and aim for just approximately optimal, then we don't need a single-parameter family: just following one of the two looping/straight-line primitives gives a lower bound of $\frac{1}{2}\sqrt{\frac{\ln k}{2\delta}}$. Let D_ℓ, D_s be the looping and straight line primitives. Then: $R_k(\mathcal{A}_{\text{single}}, \{D_\ell, D_s\}) \geq \frac{1}{2}\sqrt{\frac{\ln k}{2\delta}} \geq \frac{1}{2} R_k(\mathcal{A}_{\text{single}}, \mathcal{D}_{\text{univ}})$. Note that while this lower bound doesn't precisely match the upper bound $R_k(\mathcal{A}_{\text{single}}, \mathcal{D}_{\text{univ}})$, the latter is at most[5] $\sqrt{\frac{\ln k}{2\delta}}$, which is at most a factor 2 larger than lower bound.

[5] This is a simple extension of the standard proof that MWA has a regret upper bound of $\sqrt{(T \ln k)/2}$ in

▶ Remark. To give a sense that the primitives offer enough variety in combination, here is a simple modification over the adversary D_{lsrand}, that we call $D_{lsrand++}$: use D_{lsrand} with probability p, and with probability $1-p$ play the looping primitive D_l for all the T steps. This increases the lower bound from $\frac{2}{3}\sqrt{(T\ln k)/2}$ to $0.68\sqrt{(T\ln k)/2}$ (see the full version [11] for this result). We believe that this can be increased further by picking the stopping time for looping from a non-uniform distribution etc.

1.3 Motivation and discussion

In this work we seek to understand the structure of worst case input sequences for a broad family of algorithms and crisply expose their vulnerabilities. By identifying such structures, we also get the precise regret suffered by them. Our motivation in exploring this question includes the following.

1. After 25 years since MWA was introduced [19, 27], we do not have a sharp regret bound for it. $\mathcal{A}_{\text{single}}$ is known to suffer a regret of at most $\sqrt{\frac{T\ln k}{2}}$, but the best known lower bound on regret is $\frac{1}{4}\sqrt{T\log_2 k}$ [13], with a factor 2.35 gap between these two bounds. For larger families like $\mathcal{A}_{\text{arb}}$, $\mathcal{A}_{\text{rand}}$ no lower bounds were known. For an algorithm as widely used as MWA, it is fruitful to have a sharp regret characterization.
2. The patterns in the worst-case adversarial sequences that we characterize are simple to spot if they exist (or even if anything close exists), and make simple amends to the algorithm that result in significant gains.
3. The problem is theoretically clean and challenging: how powerful are simple input patterns beyond the typically used pure random sequences in inflicting regret?

Related Work

Classic works: The book by Cesa-Bianchi and Lugosi [7] is an excellent source for both applications and references for prediction with expert advice. The prediction with experts advice paradigm was introduced by Littlestone and Warmuth [19] and Vovk [27]. The famous multiplicative weights update algorithm was introduced independently by these two works: as the weighted majority algorithm by [19] and as the aggregating algorithm by [27]. The pioneering work of Cesa-Bianchi et al. [6] considered $\{0,1\}$ outcome space for nature and showed that for the absolute loss function $\ell(x,y)=|x-y|$ (or $g(x,y)=1-|x-y|$), the asymptotically optimal regret is $\sqrt{\frac{T\ln k}{2}}$. This was later extended to $[0,1]$ outcomes for nature by Haussler et al. [15]. The asymptotic optimality of $\sqrt{\frac{T\ln k}{2}}$ for arbitrary loss (gain) functions follows from the analysis of Cesa-Bianchi [5]. When it is known beforehand that the cumulative loss of the optimal expert is going to be small, the optimal regret can be considerably improved, and such results were obtained by Littlestone and Warmuth [19] and Freund and Schapire [9]. With certain assumptions on the loss function, the simplest possible algorithm of following the best expert already guarantees sub-linear regret (see Hannan [14]). Even when the loss functions are unbounded, if the loss functions are exponential concave, sub-linear regret can still be achieved as shown by Blum and Kalai [4]. While our work is focused on the worst-case adversaries for the MWA family of algorithms, Freund and Schapire [10] prove lower bounds for general adaptive game-playing algorithms.

the finite horizon setting with T steps and k experts, and the realization that in the geometric horizon setting, the expected stopping time is $\frac{1}{\delta}$.

Recent works: Gravin et al. [12] give the minimax optimal algorithm, and the regret for the prediction with expert advice problem for the cases of $k = 2$ and $k = 3$ experts. The focus of [12] was providing a regret upper bound for the family of all algorithms, while the focus of this paper is to provide regret lower bounds for large families of algorithms. Luo and Schapire [20] consider a setting where the adversary is restricted to pick gain vectors from the basis vector space $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$. Abernethy et al. [3] consider a different variant of experts problem where the game stops when cumulative loss of any expert exceeds given threshold. Abernethy et al. [2] consider general convex games and compute the minimax regret exactly when the input space is a ball, and show that the algorithms of Zinkevich [28] and Hazan et al. [16] are optimal w.r.t. minimax regret. Abernethy et al. [1] provide upper and lower bounds on the regret of an optimal strategy for several online learning problems without providing algorithms, by relating the optimal regret to the behavior of a certain stochastic process. Mukherjee and Schapire [23] consider a continuous experts setting where the algorithm knows beforehand the maximum number of mistakes of the best expert. Rakhlin et al. [25] introduce the notion of sequential Rademacher complexity and use it to analyze the learnability of several problems in online learning w.r.t. minimax regret. Rakhlin et al. [26] use the sequential Rademacher complexity introduced in [25] to analyze learnability w.r.t. general notions of regret (and not just minimax regret). Rakhlin et al. [24] use the notion of conditional sequential Rademacher complexity to find relaxations of problems like prediction with static experts that immediately lead to algorithms and associated regret guarantees. They show that the random playout strategy has a sound basis and propose a general method to design algorithms as a random playout. Koolen [17] studies the regret w.r.t. every expert, rather than just the best expert in hindsight and considers tradeoffs in the Pareto-frontier. McMahan and Abernethy [21] characterize the minimax optimal regret for online linear optimization games as the supremum over the expected value of a function of a martingale difference sequence, and similar characterizations for the minimax optimal algorithm and the adversary. McMahan and Orabona [22] study online linear optimization in Hilbert spaces and characterize minimax optimal algorithms. Chaudhuri et al. [8] describe a parameter-free learning algorithm motivated by the cases of large number of experts k. Koolen and Erven [18] develop a prediction strategy called Squint, and prove bounds that incorporate both quantile and variance guarantees.

2 Finite horizon

We begin our analysis of MWA by focusing on the simple case of $k = 2$ experts. We first identify the structure of the optimal adversary, and through it we obtain the tight regret bound as $T \to \infty$. Before proceeding further, it is useful to recall that when the gains of the leading and lagging experts are given by $g + d$ and g, the MWA algorithm follows these experts with probabilities $\frac{e^{\eta d}}{e^{\eta d}+1}$ and $\frac{1}{e^{\eta d}+1}$ respectively. Thus, when the adversary increases d by 1 i.e., increases the gain of the leading expert by 1, the regret benchmark (namely, the gains of the leading expert) increases by 1, where as MWA is correct only with probability $\frac{e^{\eta d}}{e^{\eta d}+1}$, and this therefore inflicts a regret of $\frac{1}{e^{\eta d}+1}$ on MWA. On the other hand, if the adversary decreases d by 1, then the benchmark doesn't change, where as MWA succeeds with probability $\frac{1}{e^{\eta d}+1}$, and this therefore inflicts a regret of $\frac{-1}{e^{\eta d}+1}$. When the adversary doesn't change d, the regret inflicted is 0.

Structure of the optimal adversary

Let η be the fixed update rate of the optimal MWA (the parameter in the exponent as explained in Section 1)[6]. Against a specific algorithm, an optimal adversary can always be found in the class of deterministic adversaries. The actions of the optimal adversary (against a specific MWA algorithm) depend only on the distance d between leading and lagging experts and time step t.

1. **Loop aggregation:** At each time step, the adversary may either increase or decrease the gap d by 1, or leave d unchanged. We denote these actions of the adversary by $d \xrightarrow{t} d+1$, $d \xrightarrow{t} d-1$, and $d \xrightarrow{t} d$. The respective regret values inflicted on the algorithm are given by $\frac{1}{e^{\eta d}+1}$, $\frac{-1}{e^{\eta d}+1}$, and 0, which are all independent of the time when an action was taken. This means that if the adversary loops between d and $d+1$ at several disconnected points of time, it may as well aggregate all of them and complete all of them in consecutive time steps. I.e., the optimal adversary starts at $d=0$ and then weakly monotonically increases d, stopping at various points $d=s$, looping for an arbitrary length of time between $d=s$ and $d=s-1$ and then proceeding forward.
2. **Staying at same d is dominated:** It is not hard to see that any action $x \to x$ is dominated for the adversary as this wastes a time step and inflicts 0 regret on the algorithm. Thus the "weakly monotonically increases" in the previous paragraph can be replaced by "strictly monotonically increases" (except of course for the stopping points for looping).
3. **Loop(0) domination:** Define $\mathrm{Loop}(d) \stackrel{\mathrm{def}}{=} d \to d+1 \to d$. It is easy to see that the regret inflicted by $\mathrm{Loop}(d)$ is exactly $\frac{1}{e^{\eta d}+1} - \frac{1}{e^{\eta(d+1)}+1}$ and this quantity is maximized at $d=0$. Thus, the optimal adversary should replace all loops by loops at 0. This gives us the structure claimed in Figures 1, 2 for the optimal adversary.

Given the optimal adversary's structure (as described in Figures 1, 2) w.l.o.g. we can assume it to be looping for $\frac{T-\ell}{2}$ steps at 0 and then monotonically increasing d for ℓ steps at which point the game ends. In the following we will analyze the regret inflicted by the optimal adversary (which we showed was optimal for the class of algorithm $\mathcal{A}_{\mathrm{single}}$) against a broader class $\mathcal{A}_{\mathrm{dec}}$ of MWA. The regret of the adversary is:

$$\sum_{t=1}^{\frac{T-\ell}{2}} \left[\frac{1}{2} - \frac{1}{e^{\eta(2t)}+1}\right] + \sum_{d=0}^{\ell-1} \frac{1}{e^{\eta(T-\ell+d+1)d}+1}. \tag{1}$$

Asymptotic regret of the optimal adversary

We first notice that for a fixed adversary with a given ℓ, the regret of MWA with decreasing $\eta(t)$ in (1) is greater than or equal to the regret of MWA with a constant $\eta' = \eta(T-\ell)$, i.e.,

$$\frac{T-\ell}{2}\left[\frac{1}{2} - \frac{1}{e^{\eta'}+1}\right] + \sum_{d=0}^{\ell-1} \frac{1}{e^{d\eta'}+1}. \tag{2}$$

This is true as each individual term in (2) is equal to or smaller than the corresponding term in (1). In the following we are going to use $\ell = T^{3/4}$ for the adversary and for convenience, we write $e^{\eta'(T-\ell)} = \tau = 1 + \frac{\alpha}{\sqrt{T}}$. The two terms in (1) together place strong bounds on what α should be: they imply that $\alpha = \Theta(1)$. We show this in 2 steps: first we show that $\alpha = O(1)$, and then show that $\alpha = \Omega(1)$.

[6] In fact, we can identify the optimal adversary for a much broader family of algorithms (see the full version [11] for more details).

1. The first term in (2) forces α to be $O(1)$. The regret of MWA for $\ell = T^{3/4}$ is at least

$$\frac{T-\ell}{2}\left[\frac{1}{2}-\frac{1}{e^{\eta'}+1}\right] \simeq \frac{T}{2}\left[\frac{1}{2}-\frac{1}{e^{\eta'}+1}\right] = \frac{\alpha\sqrt{T}}{4(1+e^{\eta'})} = \frac{\sqrt{T}}{4(\frac{2}{\alpha}+\frac{1}{\sqrt{T}})}.$$

   Since MWA's regret upper bound in the finite horizon model is $\Theta\left(\sqrt{T}\right)$, α must be $O(1)$.

2. To show that $\alpha = \Theta(1)$ we argue that the regret from the second term of (2) is $\omega(\sqrt{T})$ when $\alpha = o(1)$. For all $d \le \frac{\sqrt{T}}{\alpha}$, we have $\tau^d = (1+\frac{\alpha}{\sqrt{T}})^d \le e$. Thus MWA's regret for $k = \min(\frac{\sqrt{T}}{\alpha}, T^{3/4})$ is at least

$$\sum_{d=0}^{k-1}\frac{1}{\tau^d+1} \ge \sum_{d=0}^{k-1}\frac{1}{e+1} = \Omega(k) = \omega\left(\sqrt{T}\right).$$

   Since MWA's regret upper bound in the finite horizon model is $\Theta\left(\sqrt{T}\right)$, we get $\alpha = \Omega(1)$.

Now, we obtain the following asymptotic estimate for the second part of (2), where $\eta' \sim e^{\eta'} - 1 = \frac{\alpha}{\sqrt{T}}$.

$$\sum_{d=0}^{\ell-1}\frac{1}{\tau^d+1} \sim \int_0^{\ell}\frac{\mathrm{d}x}{e^{\eta' x}+1} = \frac{1}{\eta'}\ln\left(2\frac{e^{\ell\eta'}}{e^{\ell\eta'}+1}\right) \sim \frac{\sqrt{T}}{\alpha}\left(\ln(2)-\ln(1+e^{-\ell\eta'})\right). \tag{3}$$

The first part of (2) can be estimated as follows

$$\frac{T-\ell}{2}\left[\frac{1}{2}-\frac{1}{e^{\eta'}+1}\right] \sim \frac{T}{2}\left[\frac{e^{\eta'}-1}{2(e^{\eta'}+1)}\right] \sim \frac{T}{2}\cdot\frac{\eta'}{4} = \frac{\alpha\sqrt{T}}{8}. \tag{4}$$

As $e^{-\ell\eta'} = e^{-\alpha T^{1/4}} = o(1)$, (3) simplifies to $\frac{\ln(2)\sqrt{T}}{\alpha}$, while the estimate for (4) is $\frac{\alpha\sqrt{T}}{8}$. Now the estimate $\sqrt{T}\left[\frac{\ln(2)}{\alpha}+\frac{\alpha}{8}\right]$ for the regret in (2) is minimized for the choice of parameter $\alpha = \sqrt{8\ln(2)}$. Then the regret of the optimal MWA is at least $\sqrt{T\cdot\frac{\ln(2)}{2}}(1+o(1))$. It is known that there is MWA for $k=2$ experts with regret at most $\sqrt{T\cdot\frac{\ln(2)}{2}}$ (asymptotic in T). Thus, we obtain the following claim 1 (in the claim below, by "optimal MWA" we mean the MWA with the optimally tuned $\eta(t) = \eta'$).

▶ **Claim 1.** *For $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{single}, \mathcal{A}_{dec}$:* $R_2(\mathcal{A}_{\mathcal{F}}, D_{lsdet}) = \sqrt{\frac{T\ln 2}{2}} = R_2(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{univ})$.

We generalize the adversary for $k=2$ and obtain a tight lower bound for $\mathcal{A}_{\mathrm{dec}}$ matching the known upper bound for arbitrary *even* number k of experts and almost matching bound for *odd* number k of experts. Since $R_k(\mathcal{A}_{\mathrm{single}}) \ge R_k(\mathcal{A}_{\mathrm{dec}})$, the lower bound in Theorem 2 below applies to $\mathcal{A}_{\mathrm{single}}$ as well.

▶ **Theorem 2.** *For $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{single}, \mathcal{A}_{dec}$:*

(i) *For even k:* $R_k(\mathcal{A}_{\mathcal{F}}, D_{lsdet}) = \sqrt{\frac{T\cdot\ln k}{2}} = R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{univ})$.

(ii) *For odd k:* $R_k(\mathcal{A}_{\mathcal{F}}, D_{lsdet}) \ge \sqrt{\frac{T\cdot\ln k}{2}\left(1-\frac{1}{k^2}\right)} \ge \sqrt{1-\frac{1}{k^2}}R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{univ})$.

Proof. Let $\eta(t)$ be the update rate of the optimal MWA, we define $\ell = T^{3/4}$ and $\eta' = \eta(T-\ell)$. We employ the following adversary for the even k number of experts:

1. Divide all experts into two equal parties, numbered A and B. For the first $\frac{T-\ell}{2}$ rounds ($\ell = T^{3/4}$), advance all the experts in party A in even numbered rounds, and all experts in party B in odd numbered rounds.
2. For the remaining ℓ steps, pick an arbitrary expert and keep advancing just that expert.

Similar to (1) this adversary obtains the regret of at least $\sum_{t=1}^{\frac{T-\ell}{2}} \left[\frac{1}{2} - \frac{1}{e^{\eta(2t)}+1}\right] + \sum_{d=0}^{\ell-1} \frac{k-1}{e^{d\cdot\eta(T-\ell+d+1)}+k-1}$. We further notice that similar to (2) the regret of MWA with decreasing $\eta(t)$ in the above expression is greater than or equal to the regret of MWA with a constant $\eta' = \eta(T-\ell)$, i.e., the previous expression is at least

$$\frac{T-\ell}{2}\left[\frac{1}{2} - \frac{1}{e^{\eta'}+1}\right] + \sum_{d=0}^{\ell-1} \frac{k-1}{e^{d\cdot\eta'}+k-1}. \tag{5}$$

We use (4) to estimate the first term of (5). We estimate the second term of (5) similar to (3) as follows.

$$\sum_{d=0}^{\ell-1} \frac{k-1}{e^{d\eta'}+k-1} \sim \int_0^{\ell} \frac{(k-1)\mathrm{d}x}{e^{x\eta'}+k-1} = \frac{1}{\eta'}\ln\left(\frac{k\cdot e^{\ell\eta'}}{e^{\ell\eta'}+k-1}\right) \sim \frac{\sqrt{T}\ln(k)}{\alpha}. \tag{6}$$

Now, combining these two estimates the regret from (5) is at least

$$\frac{\alpha\sqrt{T}}{8} + \frac{\sqrt{T}\ln(k)}{\alpha} \geq 2\cdot\sqrt{\frac{\alpha\sqrt{T}}{8}\cdot\frac{\sqrt{T}\ln(k)}{\alpha}} = \sqrt{\frac{T\ln(k)}{2}},$$

which precisely matches the upper bound on the regret of MWA[6].

For the odd k number of experts we employ almost the same adversary as for even k, although, since k now is odd, we split experts into two parties of *almost* equal sizes (see the full version [11] for more details). ◀

2.1 General variations of MWA

We have seen that the best known MWA with a flat learning rate η achieves optimal (or almost optimal in the case of odd number of experts) regret among all MWAs with monotone decreasing learning rates $\eta(t)$. However, it seems that in the finite horizon model a better strategy for tuning parameters of MWA would be to use higher rates $\eta(t)$ towards the end T. In the following we study a broader family of MW algorithms $\mathcal{A}_{\text{arb}}$ where learning parameter $\eta(t)$ can vary in an arbitrary way. In the following theorem we show that such adaptivity of MWA cannot decrease the regret of the algorithm by more than a factor of 2/3.

▶ Remark. In fact, our analysis extends to the family $\mathcal{A}_{\text{rand}}$ where each $\eta(t)$ can be a random variable drawn from a distribution F_t. Effectively, with a random $\eta(t)$ the algorithm player can get any convex combination $f(G_{i,t-1}, t) = \mathbf{E}_{\eta(t)}[e^{\eta(t)G_{i,t-1}}]$ of $e^{\eta(t)G_{i,t-1}}$ in the vector of probabilities for following each expert i at time t. This constitutes a much richer family of algorithms compared to the standard single parameter MWA family.

▶ **Theorem 3.** *For $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{arb}, \mathcal{A}_{rand}$:*

(i) *For even k: $R_k(\mathcal{A}_{\mathcal{F}}, D_{lsrand}) \geq \frac{2}{3}\sqrt{\frac{T\cdot\ln k}{2}} \geq \frac{2}{3}R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{univ})$.*

(ii) *For odd k: $R_k(\mathcal{A}_{\mathcal{F}}, D_{lsrand}) \geq \frac{2}{3}\sqrt{\frac{T\cdot\ln k}{2}\left(1-\frac{1}{k^2}\right)} \geq \frac{2}{3}\sqrt{1-\frac{1}{k^2}}R_k(\mathcal{A}_{\mathcal{F}}, \mathcal{D}_{univ})$.*

Proof. Define $\ell = T^{3/4}$ and $R = \left[\frac{T-\ell-1}{2}\right]$. We use the following adversary for even number of experts k:

1. Choose $j \in [R]$ uniformly at random. With probability 0.5 don't advance any expert in the first step.
2. Divide all experts into two equal parties, numbered A and B. For the next j rounds, advance all the experts in party A in even numbered rounds, and all experts in party B in odd numbered rounds.
3. For next ℓ steps, pick any expert i and keep advancing just expert i. Do nothing in remaining steps.

The regret of the algorithm is

$$\frac{1}{R}\sum_{j=0}^{R-1}\left[\frac{1}{2}\left(\sum_{t=1}^{j}\left[\frac{1}{2}-\frac{1}{e^{\eta(2t)}+1}\right]+\sum_{d=0}^{\ell-1}\frac{k-1}{e^{d\cdot\eta(2j+d+1)}+k-1}\right)+ \frac{1}{2}\left(\sum_{t=1}^{j}\left[\frac{1}{2}-\frac{1}{e^{\eta(2t+1)}+1}\right]+\sum_{d=0}^{\ell-1}\frac{k-1}{e^{d\cdot\eta(2j+d+2)}+k-1}\right)\right]. \tag{7}$$

Since $\eta(t)$ can be arbitrary nonnegative number, we break (7) into terms with the same $\eta(t)$ (we also drop a few terms to simplify the expression). In the following, we will also assume that $e^{\eta(t)} = 1 + \frac{\alpha(t)}{\sqrt{T}}$, where $\alpha(t) = \Theta(1)$ for every $t \in [T]$. Later we will explain why this assumption is without loss of generality.

$$\begin{aligned}(7) &\geq \frac{1}{2R}\sum_{t=\ell}^{T-\ell-1}\left(\left[R-\lceil t/2\rceil\right]\cdot\left[\frac{1}{2}-\frac{1}{e^{\eta(t)}+1}\right]+\sum_{d=0}^{\ell-1}\frac{k-1}{e^{\eta(t)d}+k-1}\right)\\ &\simeq \sum_{t=\ell}^{T-\ell-1}\left(\left[\frac{R-\lceil t/2\rceil}{R}\right]\frac{\alpha(t)}{8\sqrt{T}}+\frac{\sqrt{T}\ln(k)}{2R\cdot\alpha(t)}\right) \geq \sum_{t=\ell}^{T-\ell-1}2\sqrt{\left[\frac{R-\lceil t/2\rceil}{R}\right]\frac{\alpha(t)}{8\sqrt{T}}\frac{\sqrt{T}\ln(k)}{2R\cdot\alpha(t)}}\\ &= \sqrt{\frac{\ln(k)}{2\cdot 2R}}\sum_{t=\ell}^{T-\ell-1}\sqrt{\left[\frac{R-\lceil t/2\rceil}{R}\right]} \simeq \sqrt{\frac{\ln(k)}{2\cdot T}}\int_0^1\sqrt{1-x}\quad \mathrm{d}x = \sqrt{\frac{\ln(k)}{2T}}\cdot\frac{2}{3}.\end{aligned} \tag{8}$$

In the above derivation we obtain the first approximation $\simeq$ by using approximations from (4) and (6).

We now argue that the assumption $\alpha(t) = \Theta(1)$ is without loss of generality for every $t \in [T]$. We apply a similar argument as in Theorem 1, but now for each individual term with a particular $\eta(t)$. The term $\sum_{d=0}^{\ell-1}\frac{k-1}{e^{\eta(t)d}+k-1}$ in (8) is already large enough for the estimate when $\alpha(t) = o(1)$. The term $\left[R-\lceil t/2\rceil\right]\cdot\left[\frac{1}{2}-\frac{1}{e^{\eta(t)}+1}\right]$ also places a strong bound of $O(1)$ on $\alpha(t)$, when $\left[R-\lceil t/2\rceil\right]$ is constant fraction of T. To argue about t close to the threshold T, we can slightly modify the adversary by playing with a small constant probability ε entirely "looping" strategy (without "straight line" part). This would make the coefficient in front of $\left[\frac{1}{2}-\frac{1}{e^{\eta(t)}+1}\right]$ to be sufficiently large, and at the same time would decrease the lower bound by at most $1-\varepsilon$ factor. Taking ε arbitrary small we obtain the bound in (8). This concludes the proof for the even number of experts

For the odd number of experts k. We slightly modify the adversary analogous to the case of odd number of experts in Theorem 2. This gives us an additional factor of $1-\frac{1}{k^2}$ for each of the looping terms.

Note that since our adversary is independent of how the $\eta(t)$'s are picked in the algorithm, it follows immediately that our lower bound applies for $\mathcal{A}_{\mathcal{F}} = \mathcal{A}_{\mathrm{arb}}, \mathcal{A}_{\mathrm{rand}}$. ◀

▶ Remark. One can slightly improve the lower bound in Theorem 3 and get a better factor than $\frac{2}{3}$. To this end we employ a more complicated adversary by playing with some probability $p > 0$ the same strategy as in Theorem 3 and with the remaining $1 - p$ probability playing purely looping strategy (see the full version [11]).

3 Geometric horizon

We prove two main results in this Section. We derive the structure of the optimal adversary for 2 experts and show that the optimal regret for 2 experts is exactly $\frac{0.391}{\sqrt{\delta}}$ as $\delta \to 0$. For an arbitrary number of experts k, we derive a regret lower bound of $\frac{1}{2}\sqrt{\frac{\ln(k)}{2\delta}}$ (see the full version [11] for proofs of these results).

References

1. Jacob Abernethy, Alekh Agarwal, Peter L. Bartlett, and Alexander Rakhlin. A stochastic view of optimal regret through minimax duality. In *COLT 2009 – The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.
2. Jacob Abernethy, Peter L. Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal stragies and minimax lower bounds for online convex games. In *21st Annual Conference on Learning Theory – COLT 2008, Helsinki, Finland, July 9-12, 2008*, pages 415–424, 2008.
3. Jacob Abernethy, Manfred K. Warmuth, and Joel Yellin. When random play is optimal against an adversary. In *COLT*, pages 437–446, 2008.
4. Avrim Blum and Adam Kalai. Universal portfolios with and without transaction costs. *Machine Learning*, 35(3):193–205, June 1999.
5. Nicolò Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, COLT'97, pages 163–170, New York, NY, USA, 1997. ACM.
6. Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, May 1997.
7. Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.
8. Kamalika Chaudhuri, Yoav Freund, and Daniel J. Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 297–305, 2009.
9. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997.
10. Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.
11. Nick Gravin, Yuval Peres, and Balasubramanian Sivan. Tight lower bounds for multiplicative weights algorithmic families, 2016. CoRR, abs/1607.02834. URL: `http://arxiv.org/abs/1607.02834`.
12. Nick Gravin, Yuval Peres, and Balasubramanian Sivan. Towards optimal algorithms for prediction with expert advice. In *To appear in the Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2016.
13. András Gyorgy, Dávid Pál, and Csaba Szepesvári. *Online Learning: Algorithms for Big Data*. Manuscript, 2013.
14. James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

15 David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Tight worst-case loss bounds for predicting with expert advice. In *EuroCOLT*, pages 69–83, 1995.

16 Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*, pages 499–513, 2006.

17 Wouter M. Koolen. The pareto regret frontier. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 863–871, 2013.

18 Wouter M. Koolen and Tim van Erven. Second-order quantile methods for experts and combinatorial games. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 1155–1175, 2015.

19 Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, February 1994.

20 Haipeng Luo and Robert E. Schapire. Towards minimax online learning with unknown time horizon. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 226–234, 2014.

21 H. Brendan McMahan and Jacob Abernethy. Minimax optimal algorithms for unconstrained linear optimization. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2724–2732, 2013.

22 H. Brendan McMahan and Francesco Orabona. Unconstrained online linear learning in hilbert spaces: Minimax algorithms and normal approximations. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, pages 1020–1039, 2014.

23 Indraneel Mukherjee and Robert E. Schapire. Learning with continuous experts using drifting games. *Theor. Comput. Sci.*, 411(29-30):2670–2683, 2010.

24 Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Relax and randomize : From value to algorithms. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2150–2158, 2012.

25 Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Random averages, combinatorial parameters, and learnability. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 1984–1992, 2010.

26 Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Beyond regret. In *COLT 2011 – The 24th Annual Conference on Learning Theory, June 9-11, 2011, Budapest, Hungary*, pages 559–594, 2011.

27 Volodimir G. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, COLT'90, pages 371–386, 1990.

28 Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 928–936, 2003.

The Power of Shared Randomness in Uncertain Communication*

Badih Ghazi[1] and Madhu Sudan[2]

1 **MIT, CSAIL, Cambridge, MA, USA**
badih@mit.edu
2 **Harvard John A. Paulson School of Engineering and Applied Sciences, Cambridge, MA, USA**
madhu@cs.harvard.edu

Abstract

In a recent work (Ghazi et al., SODA 2016), the authors with Komargodski and Kothari initiated the study of *communication with contextual uncertainty*, a setup aiming to understand how efficient communication is possible when the communicating parties imperfectly share a huge context. In this setting, Alice is given a function f and an input string x, and Bob is given a function g and an input string y. The pair (x, y) comes from a known distribution μ and f and g are guaranteed to be close under this distribution. Alice and Bob wish to compute $g(x, y)$ with high probability. The lack of agreement between Alice and Bob on the function that is being computed captures the uncertainty in the context. The previous work showed that any problem with one-way communication complexity k in the standard model (i.e., without uncertainty[1]) has *public-coin* communication at most $O(k(1 + I))$ bits in the uncertain case, where I is the mutual information between x and y. Moreover, a lower bound of $\Omega(\sqrt{I})$ bits on the public-coin uncertain communication was also shown.

However, an important question that was left open is related to the power that public randomness brings to uncertain communication. Can Alice and Bob achieve efficient communication amid uncertainty without using public randomness? And how powerful are public-coin protocols in overcoming uncertainty? Motivated by these two questions:

- We prove the first separation between private-coin uncertain communication and public-coin uncertain communication. Namely, we exhibit a function class for which the communication in the standard model and the public-coin uncertain communication are $O(1)$ while the private-coin uncertain communication is a growing function of n (the length of the inputs). This lower bound (proved with respect to the uniform distribution) is in sharp contrast with the case of public-coin uncertain communication which was shown by the previous work to be within a constant factor from the certain communication. This lower bound also implies the first separation between public-coin uncertain communication and deterministic uncertain communication. Interestingly, we also show that if Alice and Bob *imperfectly share* a sequence of random bits (a setup weaker than public randomness), then achieving a constant blow-up in communication is still possible.
- We improve the lower-bound of the previous work on public-coin uncertain communication. Namely, we exhibit a function class and a distribution (with mutual information $I \approx n$) for which the one-way certain communication is k bits but the one-way public-coin uncertain communication is at least $\Omega(\sqrt{k} \cdot \sqrt{I})$ bits.

Our proofs introduce new problems in the standard communication complexity model and prove lower bounds for these problems. Both the problems and the lower bound techniques may be of general interest.

* A full version of the paper is available at https://arxiv.org/abs/1705.01082 [7].
[1] In other words, under the promise that $f = g$.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 49; pp. 49:1–49:14

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

1998 ACM Subject Classification E.4 Coding and Information Theory

Keywords and phrases randomness, uncertainty, communication, imperfectly shared randomness, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.49

1 Introduction

In many forms of communication (e.g., human, computer-to-computer), the communicating parties share some context (e.g, knowledge of a language, operating system, communication protocol, encoding/decoding mechanisms.). This context is usually a) *huge* and b) *imperfectly shared* among the parties. Nevertheless, in human communication, very efficient communication is usually possible. Can we come up with a mathematical analogue of this phenomenon of efficient communication based on a huge but imperfectly shared context? Motivated by this general question, the study of "communication amid uncertainty" has been the subject of a series of recent work starting with Goldreich, Juba and Sudan [13, 8] followed by [12, 14, 15, 11, 4]. While early works were very abstract and general, later works (starting with Juba, Kalai, Khanna and Sudan [12]) tried to explore the ramifications of uncertainty in Yao's standard communication complexity model [26]. In particular, the more recent works relax the different pieces of context that were assumed to be *perfectly shared* in Yao's model, such as shared randomness [4], and in a recent work of the authors with Komargodski and Kothari the *function being computed* [6].

Specifically, [6] study the following functional notion of uncertainty in communication. Their setup builds on – and generalizes – Yao's classical model of (distributional) communication complexity, where Alice has an input x and Bob has an input y, with (x, y) being sampled from a distribution μ. Their goal is to communicate minimally so as to compute some function $g(x, y)$ (with high probability over the choice of (x, y)). The understated emphasis of the model is that for many functions g, the communication required is much less than the lengths of x or y, the entropy of x or y or even the conditional entropy of x given y.

The question studied by [6] is: How much of this gain in communication is preserved when the communicating parties do not exactly agree on the function being computed? (We further discuss the importance of this question in Section 1.2.) This variation of the problem is modelled as follows: Alice is given a Boolean function f and an input string x, and Bob is given a Boolean function g and an input string y where (x, y) is sampled from a known distribution μ as before, and (f, g) is chosen (adversarially) from a known class $\mathcal{F}$ of pairs of functions that are close in terms of the Hamming distance Δ_μ (weighted according to μ). Alice and Bob wish to compute $g(x, y)$. Alice's knowledge of the function f (which is close but not necessarily equal to g) captures the uncertainty in the knowledge of the context.

We define the *public-coin uncertain communication complexity* $\mathsf{PubCCU}^{\mu}_{\epsilon}(\mathcal{F})$ as the minimum length of a two-way public-coin protocol whose output is correct with probability at least $1-\epsilon$ over its internal randomness and that of (x, y). We similarly define the *private-coin uncertain communication complexity* $\mathsf{PrivCCU}^{\mu}_{\epsilon}(\mathcal{F})$ by restricting to private-coin protocols. Clearly, $\mathsf{PubCCU}^{\mu}_{\epsilon}(\mathcal{F}) \leq \mathsf{PrivCCU}^{\mu}_{\epsilon}(\mathcal{F})$. The quantities $\mathsf{owPubCCU}^{\mu}_{\epsilon}(\mathcal{F})$ and $\mathsf{owPrivCCU}^{\mu}_{\epsilon}(\mathcal{F})$ are similarly defined by restricting to one-way protocols.[2]

[2] Note that the uncertain model is clearly a generalization of Yao's model which corresponds to the particular case where $\mathcal{F} = \{(f, f)\}$ for some fixed function f. On the other hand, the uncertain model

The previous work ([6]) gave an upper bound on $\mathsf{owPubCCU}^{\mu}_{\epsilon}(\mathcal{F})$ whenever $\mathcal{F}$ consists of functions g whose one-way distributional complexity is small. More precisely, denote by $\mathsf{owCC}^{\mu}_{\epsilon}(g)$ the one-way communication complexity of g in the standard distributional model.[3] Namely, $\mathsf{owCC}^{\mu}_{\epsilon}(g)$ is the minimum length of a one-way deterministic protocol computing g with probability at least $1-\epsilon$ over the randomness of (x,y). Then, [6] showed that if $\mathcal{F}$ consists of pairs (f,g) of functions that are at distance δ, and if $\mathsf{owCC}^{\mu}_{\epsilon}(f), \mathsf{owCC}^{\mu}_{\epsilon}(g) \leq k$, then for every positive θ, $\mathsf{owPubCCU}^{\mu}_{\epsilon+2\delta+\theta}(\mathcal{F}) \leq O_{\theta}(k \cdot (1+I(x;y)))$, where $I(x;y)$ denotes the mutual information between x and y.[4] Note that if μ is a product distribution and if we let the parameter θ be a small constant, then the blow-up in communication is only a constant factor. However, the protocol of [6] crucially uses public randomness, and one of the main motivations behind this work is to understand how large the blow-up would be in the case where Alice and Bob have access to weaker types of randomness (or no randomness at all).

We point out that understanding the type of randomness that is needed in order to cope with uncertainty is a core question in the setup of communication with contextual uncertainty: *If Alice and Bob do not (perfectly) agree on the function being computed, why can we assume that they (perfectly) agree on the shared randomness?*

1.1 Our Contributions

We prove several results about the power of shared randomness in uncertain communication.

Private and Imperfectly Shared Randomness

Our first result (Theorem 1) shows that private-coin protocols are much weaker than public-coin protocols in the setup of communication with contextual uncertainty. Far from obtaining a constant factor blow-up in communication, private-coin protocols incur an increase that is a growing function of n when dealing with uncertainty.

Let $\mathcal{U} \triangleq \mathcal{U}_{2n}$ be the uniform distribution on $\{0,1\}^{2\cdot n}$. For positive integers t and n, we define $\log^{(t)}(n)$ by setting $\log^{(1)}(n) = \log n$, and $\log^{(i)}(n) = \max(\log\log^{(i-1)}(n), 1)$ for all $i \in \{2,\ldots,t\}$.

▶ **Theorem 1** (Lower-bound on private-coin uncertain protocols). *For every sufficiently small $\delta > 0$, there exist a positive integer $\ell \triangleq \ell(\delta)$ and a function class $\mathcal{F} \triangleq \mathcal{F}_{\delta}$ such that*

- **(i)** *For each $(f,g) \in \mathcal{F}$, we have that $\Delta_{\mathcal{U}}(f,g) \leq \delta$.*
- **(ii)** *For each $(f,g) \in \mathcal{F}$, we have that $\mathsf{owCC}^{\mathcal{U}}_{0}(f), \mathsf{owCC}^{\mathcal{U}}_{0}(g) \leq \ell$.*
- **(iii)** *For every $\eta > 0$ and $\epsilon \in (4\delta, 0.5]$, we have that $\mathsf{PrivCCU}^{\mathcal{U}}_{\epsilon/2-2\delta-\eta}(\mathcal{F}) = \Omega(\eta^2 \cdot \log^{(t)}(n))$ for some positive integer $t = \Theta((\epsilon/\delta)^2)$.*

In Theorem 1, the inputs x and y are binary strings of length n and $\mathcal{F}$ is a family of pairs of functions, which each function mapping $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}$. Also, the parameter η

can also be viewed as a particular case of Yao's model via an exponential blow-up in the input size. For more on this view (which turns out to be ineffective in our setup), we refer the reader to Note 9 at the end of this section.

[3] By the "easy direction" of Yao's min-max principle, we can without loss of generality consider deterministic (instead of public-coin) protocols when defining $\mathsf{owCC}^{\mu}_{\epsilon}(g)$. We point out that this is not true in the uncertain case.

[4] One interpretation of the dependence of the communication in the uncertain setup on the mutual information $I(x;y)$ is that the players are better able to use the correlation of their inputs in the standard case than in the uncertain case.

can possibly depend on n. We point out that Theorem 1 also implies the first separation between *deterministic* uncertain protocols and public-coin uncertain protocols[5].

▶ Note 2. We point out that the relative power of private-coin and public-coin protocols in the uncertain model is both conceptually and technically different from the standard model. Specifically, the randomness is potentially used in the standard model in order to fool an adversary selecting the input pair (x, y), whereas in the uncertain model, it is used to fool an adversary selecting the pair (f, g) of functions that are promised to be close. This promise makes the task of proving lower bounds against private-coin protocols in the uncertain model (e.g., Theorem 1) significantly more challenging than in the standard model.[6] Moreover, a well-known theorem due to Newman [21] shows that in the standard model, any public-coin protocol can be simulated by a private-coin protocol while increasing the communication by an additive $O(\log n)$ bits. By contrast, there is no known analogue of Newman's theorem in the uncertain case!

▶ Note 3. The construction that we use to prove Theorem 1 cannot give a separation larger than $\Theta(\log \log n)$. Thus, showing a separation of $\omega(\log \log n)$ between private-coin and public-coin protocols in the uncertain case would require a new construction. For more details, see Note 13.

In light of Theorem 1, it is necessary for Alice and Bob to share some form of randomness in order to only incur a constant blow-up in communication for product distributions. Fortunately, it turns out that it is not necessary for Alice and Bob to *perfectly* share a sequence of random coins. If Alice is given a uniform-random string r of bits and Bob is given a string r' obtained by independently flipping each coordinate of r with probability 0.49, then efficient communication is still possible!

More formally, for $\rho \in [0, 1]$, define $\mathsf{owIsrCCU}^{\mu}_{\epsilon,\rho}(\mathcal{F})$ in the same way that we defined $\mathsf{owPubCCU}^{\mu}_{\epsilon}(\mathcal{F})$ except that instead of Alice and Bob having access to public randomness, Alice will have access to a sequence r of independent uniformly-random bits, and Bob will have access to a sequence r' of bits obtained by independently flipping each coordinate of r with probability $(1 - \rho)/2$. Note that this setup of imperfectly shared randomness interpolates between the public randomness and private randomness setups, i.e., $\mathsf{owIsrCCU}^{\mu}_{\epsilon,1}(\mathcal{F}) = \mathsf{owPubCCU}^{\mu}_{\epsilon}(\mathcal{F})$ and $\mathsf{owIsrCCU}^{\mu}_{\epsilon,0}(\mathcal{F}) = \mathsf{owPrivCCU}^{\mu}_{\epsilon}(\mathcal{F})$.

▶ **Theorem 4** (Uncertain protocol using imperfectly shared randomness)**.** *Let $\rho \in (0, 1]$ and μ be a* product *distribution. Let $\mathcal{F}$ consist of pairs (f, g) of functions with $\Delta_\mu(f, g) \leq \delta$, and $\mathsf{owCC}^{\mu}_{\epsilon}(f), \mathsf{owCC}^{\mu}_{\epsilon}(g) \leq k$. Then, for every positive θ, $\mathsf{owIsrCCU}^{\mu}_{\epsilon+2\delta+\theta,\rho}(\mathcal{F}) \leq O_\theta(k/\rho^2)$.*

The *imperfectly shared randomness* model in Theorem 4 was recently independently introduced (in the setup of communication complexity) by Bavarian, Gavinsky and Ito [2] and by Canonne, Guruswami, Meka and Sudan [4] (and it was further studied in [5]). Moreover, our proof of Theorem 4 is based on combining the uncertain protocol of [6] and the locality-sensitive-hashing based protocol of [4].

We point out that Theorem 4 also holds for more general i.i.d. sources of correlated randomness than the one described above. More precisely, for i.i.d. (not necessarily binary)

[5] This uses the fact that private-coin communication complexity is no larger than deterministic communication complexity, both in the certain and uncertain setups.

[6] In particular, the diagonilization-based arguments that imply a separation between the public-coin and the private-coin communication complexities of the Equality function in the standard model completely fail when we impose such a promise.

sources of (imperfectly) shared randomness with *maximal correlation*[7] ρ, the work of Witsenhausen [24] along with the protocols of [4] and [6] imply an uncertain protocol with $O_\theta(k/\rho^2)$ bits of communication.

Public Randomness

We now turn to our next result where we consider the dependence of the upper bound of [6] on the mutual information $I \triangleq I(X;Y)$ in the case of public-coin protocols. The previous work [6] proved a lower bound of $\Omega(\sqrt{I})$ on this dependence, but their lower-bound does not grow with k. We improve this lower bound to $\Omega(\sqrt{k} \cdot \sqrt{I})$.

▶ **Theorem 5** (Improved lower-bound on public-coin uncertain protocols)**.** *For every sufficiently small $\delta > 0$ and every positive integers k, n such that $k = o(\exp(\sqrt{n}))$, there exist an input distribution μ on input pairs $(X, Y) \in \{0,1\}^{k \cdot n} \times \{0,1\}^{k \cdot n}$ with mutual information $I \approx k \cdot n$ and a function class $\mathcal{F} \triangleq \mathcal{F}_{\delta,k,n}$ such that*[8]

- **(i)** *For each $(f, g) \in \mathcal{F}$, we have that $\Delta_\mu(f, g) \leq \delta$.*
- **(ii)** *For each $(f, g) \in \mathcal{F}$, we have that $\mathsf{owCC}_0^\mu(f), \mathsf{owCC}_0^\mu(g) \leq k$.*
- **(iii)** $\mathsf{owPubCCU}_\epsilon^\mu(\mathcal{F}) = \Omega(\sqrt{k} \cdot \sqrt{I})$ *for some absolute constant $\epsilon > 0$ independent of δ.*

As will be explained in detail in Section 2.2, the proof of Theorem 5 is based on an extension of the lower bound construction of [6], which is then analyzed using different techniques.

▶ Note 6. The construction that we use to prove Theorem 5 cannot give a lower bound larger than $\tilde{\Theta}(\sqrt{k} \cdot \sqrt{I})$. Thus, improving on the lower bound in Theorem 5 by more than logarithmic factors in k and I would require a new construction.

New Communication Problems

Our lower bounds in Theorems 1 and 5 are derived by defining new problems in *standard* communication complexity (i.e., without uncertainty) and proving lower bounds for these problems. We describe these problems and our results on these next.

The construction that we use to prove Theorem 1 requires us to understand the following "subset-majority with side information" setup. Alice is given a subset $S \subseteq [n]$ and a string $x \in \{\pm 1\}^n$, and Bob is given a subset $T \subseteq [n]$ and a string $y \in \{\pm 1\}^n$. The subsets S and T are adversarially chosen but are promised to satisfy $S \subseteq T$, $|T| = \ell$ and $|T \setminus S| \leq \delta \cdot \ell$ for some fixed parameters ℓ and δ. The strings x and y are chosen independently and uniformly at random. Alice and Bob wish to compute the function $\mathsf{SubsetMaj}((S,x),(T,y)) \triangleq \mathsf{Sign}(\sum_{i \in T} x_i y_i)$. In words, $\mathsf{SubsetMaj}((S,x),(T,y))$ is equal to 0 if x and y differ on a majority of the coordinates in subset T, and 1 otherwise. Note that S does not directly appear in the definition of the function $\mathsf{SubsetMaj}$ but it can serve as useful side-information for Alice.[9] What is the private-coin communication complexity of computing $\mathsf{SubsetMaj}$ on every (S,T)-pair satisfying the above promise and with high probability over the random choice of (x,y) and over the private randomness? We prove the following (informally stated) lower bound.

---

[7] The *maximal correlation* of a pair (X,Y) of random variables (with support $\mathcal{X} \times \mathcal{Y}$) is defined as $\rho(X,Y) \triangleq \sup \mathbb{E}[F(X)G(Y)]$ where the supremum is over all functions $F : \mathcal{X} \to \mathbb{R}$ and $G : \mathcal{Y} \to \mathbb{R}$ with $\mathbb{E}[F(X)] = \mathbb{E}[G(Y)] = 0$ and $\mathsf{Var}[F(X)] = \mathsf{Var}[G(Y)] = 1$. It is not hard to show that the binary source of imperfectly shared randomness defined in the paragraph preceding Theorem 4 has maximal correlation ρ.

[8] We note that $I \approx k \cdot n$ means that $I/(k \cdot n) \to 0$ as $n \to \infty$.

[9] Note that we could have alternatively defined $\mathsf{SubsetMaj}$ in terms of S, and let T serve as the potentially useful side-information. Our lower bound would also apply to this setup.

▶ **Lemma 7.** *Any private-coin protocol computing* SubsetMaj *on every* (S,T)*-pair satisfying the promise and with high probability over the random choice of* (x,y) *and over the private randomness should communicate at least* $\log^{(t)}(n)$ *bits for some positive integer* t *that depends on* δ *and the error probability.*

The proof of Theorem 5 is based on a construction that leads to the question described next regarding the communication complexity of a particular block-composed function. Namely, consider the following "majority composed with subset-parity with side information" setup. Alice is given a sequence of subsets $S \triangleq (S^{(i)} \subseteq [n])_{i\in[k]}$ and a sequence of strings $x \triangleq (x^{(i)} \in \{0,1\}^n)_{i\in[k]}$, and Bob is given a sequence of subsets $T \triangleq (T^{(i)} \subseteq [n])_{i\in[k]}$ and a sequence of strings $y \triangleq (y^{(i)} \in \{0,1\}^n)_{i\in[k]}$. We consider the following distribution μ on $((S,x),(T,y))$. Independently for each $i \in [k]$, we sample $((S^{(i)},x^{(i)}),(T^{(i)},y^{(i)}))$ as follows: $S^{(i)}$ is a uniform-random subset and $T^{(i)}$ is an ϵ-noisy[10] copy of $S^{(i)}$, and independently $x^{(i)}$ is a uniform-random string and $y^{(i)}$ is an ϵ-noisy copy of x. Here, ϵ is a positive parameter that can depend on n and k. Alice and Bob wish to compute the function $\mathsf{Maj} \circ \mathsf{SubsetParity}((S,x),(T,y)) \triangleq \mathsf{Sign}\big(\sum_{i=1}^{k}(-1)^{\langle T^{(i)},x^{(i)}\oplus y^{(i)}\rangle}\big)$ where $T^{(i)}$ denotes both the subset and its 0/1 indicator vector, the inner product is over $\mathbb{F}_2$, and $x^{(i)} \oplus y^{(i)}$ is the coordinate-wise XOR of $x^{(i)}$ and $y^{(i)}$. What is the communication complexity of computing $\mathsf{Maj} \circ \mathsf{SubsetParity}$ with high probability over the distribution μ? We prove the following lower bound.

▶ **Lemma 8.** *Any* 1*-way protocol computing* $\mathsf{Maj} \circ \mathsf{SubsetParity}$ *with high probability over the distribution* μ *should communicate* $\Omega(k \cdot \epsilon \cdot n)$ *bits.*

In Section 2.1, we outline the proof of Theorem 1 and explain how it leads to the setup of Lemma 7 and how we prove Lemma 7. In Section 2.2, we outline the proof of Theorem 5 and explain how it leads to the setup of Lemma 8 and how we prove Lemma 8.

Before doing so, we discuss some conceptual implications of our results.

1.2 Implications

Functional uncertainty models much of the day-to-day interactions among humans, where a person is somewhat aware of the objectives of the other person she is interacting with, but do not know them precisely. Neither person typically knows exactly what aspects of their own knowledge may be relevant to the interaction, yet they do manage to have a short conversation. This is certainly a striking phenomenon, mostly unexplained in mathematical terms. This line of works aims to explore such phenomena. It is important to understand what mechanisms may come into play here, and what features play a role. Is the ability to make random choices important? Is shared information crucial? Is there a particular measure of distance between functions that makes efficient communication feasible? In order to understand such questions, one first needs to have a ground-level understanding of communication with functional uncertainty. This work tackles several basic questions that remain unexplored.

An ideal model for communication would only assume a constant amount of perfectly shared context between the sender and receiver, such as the knowledge of an encoding/decoding algorithm, one universal Turing machine, etc. Solutions to most interesting communication problems seem to assume a shared information which grows with the length of the inputs.

[10] This means that the indicator vector of $T^{(i)}$ is obtained by independently flipping each coordinate of the indicator vector of $S^{(i)}$ with probability ϵ.

Recent work showed that in many of these scenarios some assumptions about the shared context can be relaxed to an imperfect sharing, but these results are often brittle and break when two or more contextual elements are simultaneously assumed to be imperfectly shared. Our work raises the question of whether *imperfectly shared randomness* would be sufficient to overcome *functional uncertainty*. We show that this is indeed the case for product distributions, but the loss for non-product distributions might be much larger (for this and other open questions, we refer the reader to our conclusion Section 6). Such results highlight the delicate nature of the role of shared context in communication. They beg for a more systematic study of communication which at the very least should be able to mimic the aims, objectives and phenomena encountered in human communication.

▶ Note 9. As mentioned in Footnote 2, the uncertain model is clearly a generalization of Yao's model. Strictly speaking, the uncertain model can also be viewed as a particular case of Yao's model by regarding the function(s) that is being computed as part of the inputs of Alice and Bob, which results in an exponential blow-up in the input-size. This latter view turns out to be fruitless for our purposes. Indeed, from this perspective, all the different well-studied communication functions (such as Equality, Set Disjointness, Pointer Jumping, etc.) are regarded as special cases of one "universal function"! More importantly, this view completely blurs the distinction between the *goal* of the communication (i.e., the function to compute) and the inputs of the parties. On a technical level, it does not simplify the task of proving the lower bounds in Theorems 1 and 5 in any way since it does not capture the promise that the two functions (given to Alice and Bob) are close in Hamming distance. Thus, in the rest of this paper, we stick to the former view and use the expressions "uncertain model" and "standard model" to refer to the setups with and without uncertainty, respectively.

2 Overview of Proofs

2.1 Overview of Proof of Theorem 1

Reduction to Lemma 7

In order to prove Theorem 1, we need to devise a function class for which circumventing the uncertainty is much easier using public randomness than using private randomness. One general setup in which Bob can leverage public randomness to resolve some uncertainty regarding Alice's knowledge is the following "small-set intersection" problem. Assume that Alice is given a subset $S \subseteq [n]$, and Bob is given a subset $T \subseteq [n]$ such that T contains S and $|T| = \ell$, where we think of ℓ as being a large constant. Here, Bob knows that Alice has a subset of his own T but he is uncertain which subset Alice has. Using public randomness, a standard 1-way hashing protocol communicating $\tilde{O}(\ell)$ bits allows Bob to determine S with high probability. On the other hand, using only private randomness, the communication complexity of this task is $\Theta(\log \log n)$ bits.

With the above general setup in mind, we consider functions f_S indexed by small subsets S of coordinates on which they depend. Since we want the functions f_S and f_T to be close in Hamming distance, we enforce $|T \setminus S|$ to be small for every pair (f_S, f_T) of functions in our class, and we let each function f_S be "noise-stable". Since we want our function f_S to genuinely depend on all coordinates in S, the majority function $f_S(x, y) = \mathsf{Sign}(\sum_{i \in S} x_i y_i)$ for $x, y \in \{\pm 1\}^n$ arises as a natural choice. We also let x and y be independent uniform-random strings. In this case, it can be seen that if $|T \setminus S|$ is a small constant fraction of $|T|$, then the quadratic polynomials $\sum_{i \in S} x_i y_i$ and $\sum_{i \in T} x_i y_i$ behave like standard Gaussians with correlation close to 1, and the quadratic threshold functions $f_S(x, y)$ and $f_T(x, y)$ are thus close in Hamming distance.

Note that in the certain case, i.e., when both Alice and Bob agree on S, they can easily compute $f_S(x, y)$ by having Alice send to Bob the ℓ bits $(x_i)_{i \in S}$. Moreover, if Alice and Bob are given access to public randomness in the uncertain case, Bob can figure out S via the hashing protocol mentioned above using $\tilde{O}(\ell)$ bits of communication, which would reduce the problem to the certain case[11]. The bulk of the proof will be to lower-bound the private-coin uncertain communication. Note that by the choice of our function class and distribution, this is equivalent to proving Lemma 7.

Proof of Lemma 7

To prove Lemma 7, the high-level intuition is that a protocol solving the uncertain problem should be essentially revealing to Bob the subset S that Alice holds. Formalizing this intuition turns out to be challenging, especially that a private-coin protocol solving the uncertain problem is only required to output a *single bit* which is supposed to equal the Boolean function $f_T(x, y)$ with high probability over (x, y) and over the private randomness. In fact, this high-level intuition can be shown not to hold in certain regimes[12]. Moreover, the standard proofs that lower bound the communication of small-set intersection do not extend to lower-bound the communication complexity of f_T.

To lower-bound the private-coin communication of solving the uncertain task by a growing function of n, we consider the following *shift communication game*. Bob is given a sorted tuple $\sigma = (\sigma_1, \ldots, \sigma_t)$ of integers with $1 \leq \sigma_1 < \cdots < \sigma_t \leq n$, and Alice is either given the prefix $(\sigma_1, \ldots, \sigma_{t-1})$ of length $t-1$ of σ or the suffix $(\sigma_2, \ldots, \sigma_t)$ of length $t-1$ of σ. Bob needs to determine the input of Alice. We show that a celebrated lower bound of Linial [19] on the *chromatic number* of certain related graphs implies a lower bound of $\log^{(t+1)}(n)$ on the private-coin communication of the shift communication game. We then show that any private-coin protocol solving the uncertain task can be turned into a private-coin protocol solving the shift-communication game with a constant (i.e., independent of n) blow-up in the communication.

2.2 Overview of Proof of Theorem 5

Reduction to Lemma 8

The proof of Theorem 5 builds on the lower-bound construction of [6] which we recall next. Let μ be the distribution over pairs $(x, y) \in \{0,1\}^{2n}$ where x is uniform-random and y is an ϵ-noisy copy of x with $\epsilon = \sqrt{\delta/n}$. Then, the mutual information between x and y satisfies $I \approx n$. For each $S \subseteq [n]$, consider the function $f_S(x, y) \triangleq \langle S, x \oplus y \rangle$ where the inner product is over $\mathbb{F}_2$, $x \oplus y$ denotes the coordinate-wise XOR of x and y, and S is used to denote both the subset and its 0/1 indicator vector. Moreover, consider the class $\mathcal{F}$ of all pairs of functions (f_S, f_T) where $|S \triangle T| \leq \sqrt{\delta n}$. It can be seen that for such S and T, the distance between f_S and f_T under μ is at most δ. If Alice and Bob both know S, then Alice can send the single bit $\langle S, x \rangle$ to Bob who can then output the correct answer $\langle S, x \oplus y \rangle = \langle S, x \rangle \oplus \langle S, y \rangle$. This means that the certain communication is 1 bit. Using the well-known discrepancy method, [6] showed a lower bound of $\Omega(\sqrt{n})$ bits on the communication of the associated uncertain problem. Since in this case $I \approx n$, this in fact lower-bounds the uncertain communication by

[11] Alternatively, Alice and Bob can run the protocol of [6] which would communicate $O(\ell)$ bits.

[12] For example, for constant error probabilities, the 1-way randomized communication complexity of small-set intersection is known to be $\Theta(\ell \cdot \log(\ell))$ bits (see, e.g., [3]) whereas the public-coin protocol of [6] can compute f_T with $O(\ell)$ bits of communication.

$\Omega(\sqrt{I})$ bits. For this construction, this lower bound turns out to be tight up to a logarithmic factor.

To improve the lower-bound from $\sqrt{I}$ to $\sqrt{k} \cdot \sqrt{I}$, we consider the following "block-composed" framework. Let $\{f_{S^{(i)}}(x^{(i)}, y^{(i)}) : i \in [k]\}$ be k independent copies of the above base problem of [6] and consider computing the composed function $g\big(f_{S^{(1)}}(x^{(1)}, y^{(1)}), \ldots, f_{S^{(k)}}(x^{(k)}, y^{(k)})\big)$ for some outer function $g : \{0,1\}^k \to \{0,1\}$. For any choice of g, the certain communication of the composed function would be at most k bits. When choosing the outer function g to use in our lower bound, we thus have two objectives to satisfy. First, g has to be sufficiently *hard* in the sense that its average-case decision tree complexity with respect to the uniform distribution on $\{0,1\}^k$ should be $\Omega(k)$; otherwise, it will not be the case that the uncertain communication of computing g on k copies of the base problem is at least k times the uncertain communication of the base problem. Second, g has to be *noise stable* in order to be able to upper bound the distance between $g\big(f_{S^{(1)}}(\cdot), \ldots, f_{S^{(k)}}(\cdot)\big)$ and $g\big(f_{T^{(1)}}(\cdot), \ldots, f_{T^{(k)}}(\cdot)\big)$.

Note that setting g to be a dictator function would satisfy the noise-stability property, but it clearly would not satisfy the hardness property, as the composed function would be equal to the base function and would thus have uncertain communication $\tilde{O}(\sqrt{n})$ bits. Another potential choice of g is to set it to the parity function on k bits. This function would satisfy the hardness property, but it would strongly violate the noise stability property that is crucial to us. This leads us to setting g to the majority function on k bits, which is well-known to be noise stable, and has average-case decision-tree complexity $\Omega(k)$ with respect to the uniform distribution on $\{0,1\}^k$. In fact, the noise stability of the majority function readily implies an upper bound of $O(\sqrt{\delta})$ on the distance between any pair of composed functions that are specified by tuples of subsets $(S^{(1)}, \ldots, S^{(k)})$ and $(T^{(1)}, \ldots, T^{(k)})$ with $|S^{(i)} \triangle T^{(i)}| \leq \sqrt{\delta n}$ for each $i \in [k]$. The crux of the proof will be to lower-bound the uncertain communication of the majority-composed function by $\Omega(k\sqrt{n})$, which amounts to proving Lemma 8. Since in this block-composed framework the mutual information satisfies $I \approx kn$, this would imply the lower bound of $\Omega(\sqrt{k}\sqrt{I})$ on the uncertain communication in Part (iii) of Theorem 5.

Proof of Lemma 8

We first point out that the average-case *quantum* decision tree complexity of Maj_k with respect to the uniform distribution is $\tilde{O}(\sqrt{k})$ [1]. This implies that any communication complexity lower-bound method that extends to the quantum model cannot prove a lower bound larger than $\tilde{O}(\sqrt{k} \cdot \sqrt{n})$ on our uncertain communication[13]. In particular, we cannot solely rely on the discrepancy bound (as done in [6]), since this bound is known to lower-bound quantum communication. Similarly, the techniques of [22, 23, 18] rely on the generalized discrepancy bound (originally due to [16]) which also lower-bounds quantum communication. Moreover, the recent results of [20] only apply to product distributions (i.e., where Alice's input is independent of Bob's input) in contrast to our case where the inputs of Alice and Bob are very highly-correlated. Finally, the recent works of [9, 10] do not imply lower bounds on the average-case complexity with respect to the distribution that arises in our setup.

To circumvent the above obstacles, we use a new approach that is tailored to our setup and that is outlined next. Let Π be a 1-way protocol solving the uncertain task with high probability. We consider the information that Π reveals about the *inputs to the outer*

[13] Thus, since $I \approx k \cdot n$ in our block-composed framework, such methods cannot be used to improve the lower-bound of $\Omega(\sqrt{I})$ of [6] by more than logarithmic factors.

function, i.e., about the length-k binary string $\left(f_{S^{(1)}}(x^{(1)}, y^{(1)}), \ldots, f_{S^{(k)}}(x^{(k)}, y^{(k)})\right)$. We call this quantity the *intermediate information cost* of Π, and we argue that it is at least $\Omega(k)$ bits. To do so, we recall the Hamming distance function HD_k defined by $\mathsf{HD}_k(u, v) = 1$ if the Hamming distance between u and v is at least $k/2$ and $\mathsf{HD}_k(u, v) = 0$ otherwise. We upper bound the information complexity of computing HD_k over the uniform distribution on $\{0, 1\}^{2k}$ by the intermediate information cost of Π. We do so by giving an information-cost preserving procedure where Alice and Bob are given independent uniformly distributed u and v (respectively) and use their private and public coins in order to simulate the input distribution (X, Y) of our uncertain problem. The known 1-way lower bound of [25] on HD_k under the uniform distribution then implies that Π reveals $\Omega(k)$ bits of information to Bob about the tuple $\left(f_{S^{(1)}}(x^{(1)}, y^{(1)}), \ldots, f_{S^{(k)}}(x^{(k)}, y^{(k)})\right)$. This allows Bob to guess this tuple with probability 0.51^k. We then apply the strong direct product theorem for discrepancy of [17] which, along with the discrepancy-based lower bound on the communication of the base uncertain problem of [6], implies that Π should be communicating at least $\Omega(k\sqrt{n})$ bits.

Organization of the rest of the paper

In Section 3, we give some preliminaries. In Sections 4 and 5, we describe the proofs of Theorems 1 and 5 respectively. All the missing proofs as well as the proof of Theorem 4 appear in the full version. In Section 6, we conclude with some interesting open questions.

3 Preliminaries

For a real number x, we define $\mathsf{Sign}(x)$ to be 1 if $x \geq 0$ and 0 if $x < 0$. For a set S, we write $X \in_R S$ to indicate that X is a random variable that is uniformly distributed on S. For a positive integer n, we let $[n] \triangleq \{1, \ldots, n\}$. For a real number x, we denote $\exp(x)$ a quantity of the form $2^{\Theta(x)}$. For any two subsets $S, T \subseteq [n]$, we let $S \setminus T$ be the set of all elements of S that are not in T. We let $S \triangle T$ be the symmetric difference of S and T, i.e., the union of $S \setminus T$ and $T \setminus S$. For functions $f, g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ and any distribution μ on $\mathcal{X} \times \mathcal{Y}$, we define the distance $\Delta_\mu(f, g) \triangleq \Pr_{(x,y)\sim\mu}[f(x, y) \neq g(x, y)]$ as the Hamming distance between the values of f and g, weighted with respect to μ. If μ is the uniform distribution on $\mathcal{X} \times \mathcal{Y}$, we drop the subscript μ and denote Δ_μ by Δ. We next recall the standard communication complexity model of Yao [26]. For any function $f\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, we denote by $\mathsf{CC}(f)$, $\mathsf{owCC}(f)$, $\mathsf{PrivCC}_\epsilon(f)$ and $\mathsf{owPrivCC}_\epsilon(f)$ its two-way deterministic, one-way deterministic, two-way private-coin and one-way private-coin communication complexity respectively. For any distribution μ over $\mathcal{X} \times \mathcal{Y}$, we denote by $\mathsf{CC}^\mu_\epsilon(f)$ and $\mathsf{owCC}^\mu_\epsilon(f)$ the two-way distributional and one-way distributional communication complexity of over μ with error ϵ, respectively.

We next recall the model of communication with contextual uncertainty. For more details on this model, we refer the reader to [6]. In this setup, Alice knows a function f and is given an input x, and Bob knows a function g and is given an input y. Let $\mathcal{F} \subseteq \{f\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}\}^2$ be a family of pairs of Boolean functions with domain $\mathcal{X} \times \mathcal{Y}$, and μ be a distribution on $\mathcal{X} \times \mathcal{Y}$. We say that a public-coin (resp. private-coin) protocol Π ϵ-computes $\mathcal{F}$ over μ if for every $(f, g) \in \mathcal{F}$, we have that Π outputs the value $g(x, y)$ with probability at least $1 - \epsilon$ over the randomness of $(x, y) \sim \mu$ and over the public (resp. private) randomness of Π.

▶ **Definition 10** (Contextually Uncertain Communication Complexity)**.** Let μ be a distribution on $\mathcal{X} \times \mathcal{Y}$ and $\mathcal{F} \subseteq \{f\colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}\}^2$. The *two-way (resp. one-way) public-coin communication complexity of $\mathcal{F}$ under contextual uncertainty*, denoted $\mathsf{PubCCU}^\mu_\epsilon(\mathcal{F})$ (resp. $\mathsf{owPubCCU}^\mu_\epsilon(\mathcal{F})$), is the minimum over all two-way (resp. one-way) public-coin protocols Π

that ϵ-compute $\mathcal{F}$ over μ, of the maximum communication complexity of Π over $(f,g) \in \mathcal{F}$, (x,y) from the support of μ and settings of the public coins.

Similarly, the *two-way (resp. one-way) private-coin communication complexity of $\mathcal{F}$ under contextual uncertainty* $\mathsf{PrivCCU}^{\mu}_{\epsilon}(\mathcal{F})$ (resp. $\mathsf{owPrivCCU}^{\mu}_{\epsilon}(\mathcal{F})$) is defined by restricting to two-way (resp. one-way) private-coin protocols.

4 Construction for Private-Coin Uncertain Protocols

We now describe the construction that is used to prove Theorem 1. Each function in our universe is specified by a subset $S \subseteq [n]$ and is of the form $f_S : \{\pm 1\}^n \times \{\pm 1\}^n \to \{0,1\}$ with $f_S(X,Y) \triangleq \mathsf{Sign}(\sum_{i \in S} X_i Y_i)$ for all $X, Y \in \{\pm 1\}^n$. The function class is then defined by

$$\mathcal{F}_\delta \triangleq \{(f_S, f_T) : S \subseteq T,\ |T| = \ell \text{ and } |T \setminus S| \leq \delta' \cdot \ell\},$$

where $\delta' = \alpha \cdot \delta^2$ for some sufficiently small positive absolute constant α, and $\ell = \ell(\delta)$ is a sufficiently large function of δ. The input pair (X,Y) is drawn from the uniform distribution on $\{\pm 1\}^{2n}$. The proof of Part (i) of Theorem 1 follows from the fact that the the polynomials $\sum_{i \in S} X_i Y_i$ and $\sum_{i \in T} X_i Y_i$ behave like zero-mean Gaussians with unit-variance and correlation $\sqrt{1-\delta'}$. The proof of Part (ii) of Theorem 1 is immediate and is given in the full version for completeness. To prove Part (iii) of Theorem 1, the next definition – which is based on the graphs studied by Linial [19]– will be crucial to us.

▶ **Definition 11** (Shift Communication Game $\mathcal{G}_{m,t}$)**.** Let m and t be positive integers with $t \leq m$. In the communication problem $\mathcal{G}_{m,t}$, Bob is given a sorted tuple $\sigma = (\sigma_1, \ldots, \sigma_t)$ of distinct integers with $1 \leq \sigma_1 < \cdots < \sigma_t \leq m$. In the YES case, Alice is given the prefix $(\sigma_1, \ldots, \sigma_{t-1})$ of length $t-1$ of σ. In the NO case, Alice is given the suffix $(\sigma_2, \ldots, \sigma_t)$ of length $t-1$ of σ. Alice and Bob need to determine which of the YES and NO cases occurs.

Lemma 12 lower-bounds the private-coin communication complexity of $\mathcal{G}_{m,t}$. Its proof uses Linial's lower bound on the chromatic number of related graphs.

▶ **Lemma 12.** *There is an absolute constant c such that for every sufficiently small $\epsilon > 0$, we have that* $\mathsf{PrivCC}_{\epsilon}(\mathcal{G}_{m,t}) \geq c \cdot \log^{(t+2)}(m)$.

The proof of Part (iii) of Theorem 1 – which is the main part in the proof of Theorem 1 – is deferred to the full version.

▶ Note 13. As mentioned in Note 3, the above construction cannot give a separation larger than $\Theta(\log \log n)$. This is because using private randomness, Bob can learn the set S using $O(\log \log n)$ bits of communication (see, e.g., [3]). Additionally, Alice can send the coordinates of X indexed by the elements of S to Bob who can then compute $f_S(X,Y)$.

5 Construction for Public-Coin Uncertain Protocols

In this section, we describe the construction that is used to prove Theorem 5. We set

$$\delta' \triangleq c \cdot \delta^2 \quad \text{and} \quad \epsilon \triangleq \sqrt{\delta'/n}, \tag{$\dagger$}$$

where δ is the parameter from the statement of Theorem 5, and $c > 0$ is a small-enough absolute constant. To define our input distribution, we first define a slightly more general distribution μ_η. The support of μ_η is $\{0,1\}^{kn} \times \{0,1\}^{kn}$ and we will view the coordinates of a sample $(x,y) \sim \mu_\eta$ as $x = (x^{(i)})_{i \in [k]}$ and $y = (y^{(i)})_{i \in [k]}$ with $x^{(i)}, y^{(i)} \in \{0,1\}^n$ for all

$i \in [k]$. A sample $(x, y) \sim \mu_\eta$ is generated by letting $x \in_R \{0,1\}^{kn}$ and for all $i \in [k]$ and $j \in [n]$, independently setting $y_j^{(i)}$ to be an η-noisy copy of $x_j^{(i)}$. In other words, we set $y_j^{(i)} = x_j^{(i)}$ w.p. $1-\eta$ and $y_j^{(i)} = 1 - x_j^{(i)}$ w.p. η. Our input distribution is then $\mu_{2\epsilon-2\epsilon^2}$.

We now define our function class $\mathcal{F}_\epsilon$. Each function in our universe is specified by a sequence of subsets $S \triangleq (S^{(i)} \subseteq [n])_{i\in[k]}$ and it is of the form $f_S : \{0,1\}^{2\cdot k\cdot n} \to \{0,1\}$ with[14]

$$f_S(x,y) \triangleq \mathsf{Sign}\Big(\sum_{i\in[k]} (-1)^{\langle S^{(i)}, x^{(i)} \oplus y^{(i)}\rangle}\Big) \tag{1}$$

for all $x, y \in \{0,1\}^{k\cdot n}$, where in Eq. (1) the inner product is over $\mathbb{F}_2$, the sum is over $\mathbb{R}$ and $x^{(i)} \oplus y^{(i)}$ denotes the coordinate-wise XOR of the two length-n binary strings $x^{(i)}$ and $y^{(i)}$. The function class is then defined by $\mathcal{F}_\epsilon \triangleq \{(f_S, f_T) : |S^{(i)} \triangle T^{(i)}| \le \epsilon \cdot n \text{ for all } i \in [k]\}$.

The proof of Part (i) of Theorem 5 follows from known bounds on the noise stability of the majority function. The proof of Part (ii) is immediate. To prove Part (iii), we first define (as in [6]) a communication problem in the standard distributional model that reduces to solving the contextually-uncertain problem specified by the function class $\mathcal{F}_\epsilon$ and the distribution $\mu_{2\epsilon-2\epsilon^2}$. For distributions ϕ and ψ, we denote by $\phi \otimes \psi$ the joint distribution of a sample from ϕ and an independent sample from ψ. The new problem is defined as follows.

- **Inputs:** Alice's input is a pair (S, x) where $S \triangleq (S^{(i)} \subseteq [n])_{i\in[k]}$ and $x \in \{0,1\}^{k\cdot n}$. Bob's input is a pair (T, y) where $T \triangleq (T^{(i)} \subseteq [n])_{i\in[k]}$ and $y \in \{0,1\}^{k\cdot n}$.
- **Distribution:** Let $\mathcal{D}_q$ be the distribution on the pair (S, T) of sequences of k subsets of $[n]$, which is defined by independently setting, for each $i \in [k]$, $S^{(i)}$ to be a uniformly-random subset of $[n]$, and $T^{(i)}$ to be a q-noisy copy of $S^{(i)}$. The distribution on the inputs of Alice and Bob is then given by $\nu_\epsilon \triangleq \mathcal{D}_\epsilon \otimes \mu_{2\epsilon-2\epsilon^2}$ with $\epsilon = \sqrt{\delta'/n}$.
- **Function:** The goal is to compute the function $F : \{0,1\}^{2kn} \times \{0,1\}^{2kn} \to \{0,1\}$ defined by $F((S,x),(T,y)) = f_T(x,y) = \mathsf{Sign}\big(\sum_{i\in[k]} (-1)^{\langle T^{(i)}, x^{(i)} \oplus y^{(i)}\rangle}\big)$.

The next proposition follows from a simple application of the Chernoff bound.

▶ **Proposition 14.** *For any $\theta > 0$, $\mathsf{owPubCCU}_\theta^{\mu_{2\epsilon-2-\epsilon^2}}(\mathcal{F}_\epsilon) \ge \mathsf{owCC}_{\theta+\theta'}^{\nu_\epsilon}(F)$ with $\theta' = 2^{-\Theta(\epsilon\cdot n)}$.*

In the full version, we prove the following lower bound on $\mathsf{owCC}_\theta^{\nu_\epsilon}(F)$, which along with Proposition 14 and the settings of ϵ and δ' in (†), implies Part (iii) of Theorem 5.

▶ **Lemma 15.** *For every sufficiently small positive constant θ, $\mathsf{owCC}_\theta^{\nu_\epsilon}(F) = \Omega(k \cdot \epsilon \cdot n)$.*

6 Open Questions

As mentioned in Notes 3 and 6 in Section 1, significantly improving the bounds from Theorems 1 and 5 seems to require fundamentally new constructions, and is a very important question. Moreover, is there an analogue of the protocol in Theorem 4 for *non-product* distributions?

Another very important and intriguing open question is whether efficient communication under contextual uncertainty is possible in the multi-round setup. Namely, if k is the r-round certain communication, can we upper bound the r-round uncertain communication by some function of k, I and possibly r? Even for $r = 2$ and when the uncertain protocol is allowed

[14] We will use the symbol $S^{(i)}$ to denote both the subset of $[n]$ and its corresponding 0/1 indicator vector.

to use public randomness, no non-trivial protocols are known in this setting. On the other hand, no separations are known for this case (beyond those known for $r = 1$) even if the protocols are restricted to be deterministic.

Acknowledgements. The authors would like to thank Ilan Komargodski, Pravesh Kothari and Mohsen Ghaffari and the anonymous reviewers for very helfpul discussions and pointers.

References

1. Andris Ambainis and Ronald De Wolf. Average-case quantum query complexity. *Journal of Physics A: Mathematical and General*, 34(35):6741, 2001.
2. Mohammad Bavarian, Dmitry Gavinsky, and Tsuyoshi Ito. On the role of shared randomness in simultaneous communication. In *Automata, Languages, and Programming*, pages 150–162. Springer, 2014.
3. Joshua Brody, Amit Chakrabarti, Ranganath Kondapally, David P. Woodruff, and Grigory Yaroslavtsev. Beyond set disjointness: the communication complexity of finding the intersection. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 106–113. ACM, 2014.
4. Clément Louis Canonne, Venkatesan Guruswami, Raghu Meka, and Madhu Sudan. Communication with imperfectly shared randomness. In *Innovations in Theoretical Computer Science, ITCS*, pages 257–262, 2015.
5. Badih Ghazi, Pritish Kamath, and Madhu Sudan. Communication complexity of permutation-invariant functions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1902–1921, 2016. doi:10.1137/1.9781611974331.ch134.
6. Badih Ghazi, Ilan Komargodski, Pravesh Kothari, and Madhu Sudan. Communication with contextual uncertainty. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 2072–2085, 2016. doi:10.1137/1.9781611974331.ch144.
7. Badih Ghazi and Madhu Sudan. The power of shared randomness in uncertain communication. *arXiv preprint arXiv:1705.01082*, 2017. URL: https://arxiv.org/abs/1705.01082.
8. Oded Goldreich, Brendan Juba, and Madhu Sudan. A theory of goal-oriented communication. *J. ACM*, 59(2):8, 2012.
9. Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. In *Proceedings of the 47th Symposium on Theory of Computing (STOC). ACM*, 2015.
10. Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 49, 2015.
11. Elad Haramaty and Madhu Sudan. Deterministic compression with uncertain priors. In *Innovations in Theoretical Computer Science, ITCS*, pages 377–386, 2014.
12. Brendan Juba, Adam Tauman Kalai, Sanjeev Khanna, and Madhu Sudan. Compression without a common prior: an information-theoretic justification for ambiguity in language. In *Innovations in Computer Science, ICS*, pages 79–86, 2011.
13. Brendan Juba and Madhu Sudan. Universal semantic communication I. In *40th Annual ACM Symposium on Theory of Computing*, pages 123–132, 2008.
14. Brendan Juba and Madhu Sudan. Efficient semantic communication via compatible beliefs. In *Innovations in Computer Science, ICS*, pages 22–31, 2011.
15. Brendan Juba and Ryan Williams. Massive online teaching to bounded learners. In *Innovations in Theoretical Computer Science, ITCS*, pages 1–10, 2013.

16 Hartmut Klauck. Lower bounds for quantum communication complexity. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 288–297. IEEE, 2001.

17 Troy Lee, Adi Shraibman, and Robert Spalek. A direct product theorem for discrepancy. In *Computational Complexity, 2008. CCC'08. 23rd Annual IEEE Conference on*, pages 71–80. IEEE, 2008.

18 Troy Lee and Shengyu Zhang. Composition theorems in communication complexity. In *Automata, Languages and Programming*, pages 475–489. Springer, 2010.

19 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

20 Marco Molinaro, David P. Woodruff, and Grigory Yaroslavtsev. Amplification of one-way information complexity via codes and noise sensitivity. In *Automata, Languages, and Programming*, pages 960–972. Springer, 2015.

21 Ilan Newman. Private vs. common random bits in communication complexity. *Information processing letters*, 39(2):67–71, 1991.

22 Alexander A. Sherstov. The pattern matrix method for lower bounds on quantum communication. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 85–94. ACM, 2008.

23 Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *arXiv preprint arXiv:0710.0095*, 2007.

24 Hans S. Witsenhausen. On sequences of pairs of dependent random variables. *SIAM Journal on Applied Mathematics*, 28(1):100–113, 1975.

25 David P. Woodruff. *Efficient and private distance approximation in the communication and streaming models*. PhD thesis, Citeseer, 2007.

26 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *11h Annual ACM Symposium on Theory of Computing*, pages 209–213, 1979.

Separation of $\mathrm{AC}^0[\oplus]$ Formulas and Circuits

Benjamin Rossman[1] **and Srikanth Srinivasan**[2]

1 **Departments of Mathematics and Computer Science, University of Toronto, Toronto, Canada**
rossman@utoronto.ca

2 **Department of Mathematics, IIT Bombay, Bombay, India**
srikanth@math.iitb.ac.in

Abstract

This paper gives the first separation between the power of *formulas* and *circuits* of equal depth in the $\mathrm{AC}^0[\oplus]$ basis (unbounded fan-in AND, OR, NOT and MOD_2 gates). We show, for all $d(n) \leq O(\frac{\log n}{\log\log n})$, that there exist *polynomial-size depth-d circuits* that are not equivalent to *depth-d formulas of size $n^{o(d)}$* (moreover, this is optimal in that $n^{o(d)}$ cannot be improved to $n^{O(d)}$). This result is obtained by a combination of new lower and upper bounds for *Approximate Majorities*, the class of Boolean functions $\{0,1\}^n \to \{0,1\}$ that agree with the Majority function on 3/4 fraction of inputs.

$\mathrm{AC}^0[\oplus]$ formula lower bound. We show that every depth-d $\mathrm{AC}^0[\oplus]$ formula of size s has a 1/8-*error polynomial approximation* over $\mathbb{F}_2$ of degree $O(\frac{1}{d}\log s)^{d-1}$. This strengthens a classic $O(\log s)^{d-1}$ degree approximation for circuits due to Razborov [12]. Since the Majority function has approximate degree $\Theta(\sqrt{n})$, this result implies an $\exp(\Omega(dn^{1/2(d-1)}))$ lower bound on the depth-d $\mathrm{AC}^0[\oplus]$ formula size of all Approximate Majority functions for all $d(n) \leq O(\log n)$.

Monotone AC^0 circuit upper bound. For all $d(n) \leq O(\frac{\log n}{\log\log n})$, we give a randomized construction of depth-d monotone AC^0 circuits (without NOT or MOD_2 gates) of size $\exp(O(n^{1/2(d-1)}))$ that compute an Approximate Majority function. This strengthens a construction of formulas of size $\exp(O(dn^{\frac{1}{2(d-1)}}))$ due to Amano [1].

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases circuit complexity, lower bounds, approximate majority, polynomial method

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.50

1 Introduction

The relative power of formulas versus circuits is one of the great mysteries in complexity theory. The central question in this area is whether NC^1 (the class of languages decidable by polynomial-size Boolean formulas) is a proper subclass of P/poly (the class of languages decidable by polynomial-size Boolean circuits). Despite decades of efforts, this question remains wide open.[1] In the meantime, there has been progress on analogues of the NC^1 vs. P/poly question in certain restricted settings. For instance, in the monotone basis (with AND and OR gates only), the power of polynomial-size formulas vs. circuits was separated

[1] In this paper we focus on non-uniform complexity classes. The question of uniform-NC^1 vs. P is wide open as well.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 50; pp. 50:1–50:13

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

by the classic lower bound of Karchmer and Wigderson [8] (on the monotone formula size of st-Connectivity).

The bounded-depth setting is another natural venue for investigating the question of formula vs. circuits. Consider the elementary fact that every depth-d circuit of size s is equivalent to a depth-d formula of size at most s^{d-1}, where we measure *size* by the number of gates. This observation is valid with respect to any basis (i.e. set of gate types). In particular, we may consider the AC^0 basis (unbounded fan-in AND, OR, NOT gates) and the $\mathrm{AC}^0[\oplus]$ basis (unbounded fan-in MOD_2 gates in addition to AND, OR, NOT gates). With respect to either basis, there is a natural depth-d analogue of the NC^1 vs. P/poly question (where $d = d(n)$ is a parameter that may depend on n), namely whether every language decidable by polynomial-size depth-d circuits is decidable by depth-d formulas of size $n^{o(d)}$ (i.e. better than the trivial $n^{O(d)}$ upper bound).

It is reasonable to expect that this question could be resolved in the sub-logarithmic depth regime ($d(n) \ll \log n$), given the powerful lower bound techniques against AC^0 circuits (Håstad's Switching Lemma [5]) and $\mathrm{AC}^0[\oplus]$ circuits (the Polynomial Method of Razborov [12] and Smolensky [14]). However, because the standard way of applying these techniques does not distinguish between circuits and formulas, it is not clear how to prove quantitatively stronger lower bounds on formula size vis-a-vis circuit size of a given function. Recent work of Rossman [13] developed a new way of applying Håstad's Switching Lemma to AC^0 formulas, in order to prove an $\exp(\Omega(dn^{1/(d-1)}))$ lower bound on the formula size of the Parity function for all $d \leq O(\log n)$. Combined with the well-known $\exp(O(n^{1/(d-1)}))$ upper bound on the circuit size of Parity, this yields an asymptotically optimal separation in the power of depth-d AC^0 formulas vs. circuits for all $d(n) \leq O(\frac{\log n}{\log\log n})$, as well as a super-polynomial separation for all $\omega(1) \leq d(n) \leq o(\log n)$.

In the present paper, we carry out a similar development for formulas vs. circuits in the $\mathrm{AC}^0[\oplus]$ basis, obtaining both an asymptotically optimal separation for all $d(n) \leq O(\frac{\log n}{\log\log n})$ and a super-polynomial separation for all $\omega(1) \leq d(n) \leq o(\log n)$. Our target functions lie in the class of *Approximate Majorities*, here defined as Boolean functions $\{0,1\}^n \to \{0,1\}$ that approximate the Majority function on 3/4 fraction of inputs. First, we show how to apply the Polynomial Method to obtain better parameters in the approximation of $\mathrm{AC}^0[\oplus]$ formulas by low-degree polynomials over $\mathbb{F}_2$. This leads to an $\exp(\Omega(dn^{1/2(d-1)}))$ lower bound on the $\mathrm{AC}^0[\oplus]$ formula size of all Approximate Majority functions. The other half of our formulas vs. circuits separation comes from an $\exp(O(n^{1/2(d-1)}))$ upper bound on the $\mathrm{AC}^0[\oplus]$ circuit size of some Approximate Majority function. In fact, this upper bound is realized by a randomized construction of monotone AC^0 circuits (without NOT or MOD_2 gates). Together these upper and lower bound give our main result:

▶ **Theorem 1.**

(i) *For all* $2 \leq d(n) \leq O(\frac{\log n}{\log\log n})$*, there exist* $\mathrm{AC}^0[\oplus]$ *circuits (in fact, monotone* AC^0 *circuits) of depth* d *and size* $\mathrm{poly}(n)$ *that are not equivalent to any* $\mathrm{AC}^0[\oplus]$ *formulas of depth* d *and size* $n^{o(d)}$*.*

(ii) *For all* $\omega(1) \leq d(n) \leq o(\log n)$*, the class of languages decidable by polynomial-size depth-*d $\mathrm{AC}^0[\oplus]$ *formulas is a proper subclass of the class of languages decidable by polynomial-size depth-*d $\mathrm{AC}^0[\oplus]$ *circuits.*

Separation (i) is asymptotically optimal, in view of the aforementioned simulation of $\mathrm{poly}(n)$-size depth-d circuits by depth-d formulas of size $n^{O(d)}$. Separation (ii) resembles an analogue of $\mathrm{NC}^1 \neq \mathrm{P/poly}$ (or rather $\mathrm{NC}^1 \neq \mathrm{AC}^1$) within the class $\mathrm{AC}^0[\oplus]$. In fact, extending separation (ii) from depth $o(\log n)$ to depth $\log n$ is equivalent the separation of NC^1 and AC^1.

1.1 Proof outline

Improved polynomial approximation

The lower bound for $\mathrm{AC}^0[\oplus]$ formulas follows the general template due to Razborov [12] on proving lower bounds for $\mathrm{AC}^0[\oplus]$ circuits using low-degree polynomials over $\mathbb{F}_2$. Razborov showed that for any Boolean function $f : \{0,1\}^n \to \{0,1\}$ that has an $\mathrm{AC}^0[\oplus]$ circuit of size s and depth d, there is a randomized polynomial $\mathbf{P}$ of degree $O(\log s)^{d-1}$ that computes f correctly on each input with probability $\frac{7}{8}$ (we call such polynomials 1/8-error *probabilistic polynomials*). By showing that some explicit Boolean function f (e.g. the Majority function or the MOD_q function for q odd) on n variables does not have such an approximation of degree less than $\Omega(\sqrt{n})$ [12, 14, 15], we get that any $\mathrm{AC}^0[\oplus]$ circuit of depth d computing f must have size $\exp(\Omega(n^{1/2(d-1)}))$.

In this paper, we improve the parameters of Razborov's polynomial approximation from above for $\mathrm{AC}^0[\oplus]$ *formulas*. More precisely, for $\mathrm{AC}^0[\oplus]$ formulas of size s and depth d, we are able to construct 1/8-error probabilistic polynomials of degree $O(\frac{1}{d}\log s)^{d-1}$. (Since every depth-$d$ circuit of size s is equivalent to a depth-d formula of size at most s^{d-1}, this result implies Razborov's original theorem that $\mathrm{AC}^0[\oplus]$ circuits of size s and depth d have 1/8-error probabilistic polynomials of degree $O(\log s)^{d-1}$.)

We illustrate the idea behind this improved polynomial approximation with the special case of a balanced formula (i.e. all gates have the same fan-in) of fan-in t and depth d. Note that the size of the formula (number of gates) is $\Theta(t^{d-1})$ and hence it suffices in this case to show that it has a 1/8-error probabilistic polynomial of degree $O(\log t)^{d-1}$. We construct the probabilistic polynomial inductively. Given a balanced formula F of depth d and fan-in t, let $F_1, \ldots, F_t$ be its subformulas of depth $d-1$. Inductively, each F_i has a 1/8-error probabilistic polynomial $\mathbf{P}_i$ of degree $O(\log t)^{d-2}$ and by a standard error-reduction [10], it has a $(1/16t)$-error probabilistic polynomial of degree $O(\log t)^{d-1}$ (in particular, at any given input $x \in \{0,1\}^n$, the probability that *there exists* an $i \in [t]$ such that $\mathbf{P}_i(x) \neq F_i(x)$ is at most 1/16). Using Razborov's construction of a 1/16-error probabilistic polynomial of degree $O(1)$ for the output gate of F and composing this with the probabilistic polynomials $\mathbf{P}_i$, we get the result for balanced formulas. This idea can be extended to general (i.e. not necessarily balanced) formulas with a careful choice of the error parameter for each subformula F_i to obtain the stronger polynomial approximation result.

Improved formula lower bounds

Combining the above approximation result with known lower bounds for polynomial approximation [12, 14, 15], we can already obtain stronger lower bounds for $\mathrm{AC}^0[\oplus]$ formulas than are known for $\mathrm{AC}^0[\oplus]$ circuits. For instance, it follows that any $\mathrm{AC}^0[\oplus]$ formula of depth d computing the Majority function on n variables must have size $\exp(\Omega(dn^{1/2(d-1)}))$ for all $d \leq O(\log n)$, which is stronger than the corresponding circuit lower bound. Similarly stronger formula lower bounds also follow for the MOD_q function (q odd).

Separation between formulas and circuits

However, the above improved lower bounds do not directly yield the claimed separation between $\mathrm{AC}^0[\oplus]$ formulas and circuits. This is because we do not have circuits computing (say) the Majority function of the required size. To be able to prove our result, we would need to show that the Majority function has $\mathrm{AC}^0[\oplus]$ circuits of depth d and size $\exp(O(n^{1/2(d-1)}))$ (where the constant in the $O(\cdot)$ is independent of d). However, as far as we know, the strongest

result in this direction [9] only yields $\mathrm{AC}^0[\oplus]$ circuits of size greater than $\exp(\Omega(n^{1/(d-1)}))$,[2] which is superpolynomially larger than the upper bound.

To circumvent this issue, we change the hard functions to the class of *Approximate Majorities*, which is the class of Boolean functions that agree with Majority function on *most* inputs. While this has the downside that we no longer are dealing with an explicitly defined function, the advantage is that the polynomial approximation method of Razborov yields tight lower bounds for some functions from this class.

Indeed, since the method of Razborov is based on polynomial approximations, it immediately follows that the same proof technique also yields the same lower bound for computing Approximate Majorities. Formally, any $\mathrm{AC}^0[\oplus]$ circuit of depth d computing *any* Approximate Majority must have size $\exp(\Omega(n^{1/2(d-1)}))$. On the upper bound side, it is known from the work of O'Donnell and Wimmer [11] and Amano [1] that there *exist* Approximate Majorities that can be computed by monotone AC^0 *formulas* of depth d and size $\exp(O(dn^{1/2(d-1)}))$. (Note that the double exponent $\frac{1}{2(d-1)}$ is now the same in the upper and lower bounds.)

We use the above ideas for our separation between $\mathrm{AC}^0[\oplus]$ formulas and circuits. Plugging in our stronger polynomial approximation for $\mathrm{AC}^0[\oplus]$ formulas, we obtain that any $\mathrm{AC}^0[\oplus]$ formula of depth d computing any Approximate Majority must have size $\exp(\Omega(dn^{1/2(d-1)}))$. In particular, this implies that Amano's construction is tight (up to the universal constant in the exponent) even for $\mathrm{AC}^0[\oplus]$ formulas.

Further, we also modify Amano's construction [1] to obtain better constant-depth *circuits* for Approximate Majorities: we show that there exist Approximate Majorities that are computed by monotone AC^0 circuits of depth d of size $\exp(O(n^{1/2(d-1)}))$ (the constant in the $O(\cdot)$ is a constant independent of d).

Smaller circuits for Approximate Majority

Our construction closely follows Amano's, which in turn is related to Valiant's probabilistic construction [17] of monotone formulas for the Majority function. However, we need to modify the construction in a suitable way that exploits the fact that we are constructing *circuits*. This modification is in a similar spirit to a construction of Hoory, Magen and Pitassi [6] who modify Valiant's construction to obtain smaller monotone circuits (of depth $\Theta(\log n)$) for computing the Majority function exactly.

At a high level, the difference between Amano's construction and ours is as follows. Amano constructs random formulas F_i of each depth $i \leq d$ as follows. The formula F_1 is the AND of a_1 independent and randomly chosen variables. For even (respectively odd) $i > 1$, F_i is the OR (respectively AND) of a_i independent and random copies of F_{i-1}. For suitable values of $a_1, \ldots, a_d \in \mathbb{N}$, the random formula F_d computes an Approximate Majority with high probability. In our construction, we build a depth i circuit C_i for each $i \leq d$ in a similar way, except that each C_i now has M different outputs. Given such a C_{i-1}, we construct C_i by taking M independent randomly chosen subsets $T_1, \ldots, T_M$ of a_i many outputs of C_{i-1} and adding gates that compute either the OR or AND (depending on whether i is even or odd) of the gates in T_i. Any of the M final gates of C_d now serves as the output gate. By an analysis similar to Amano's (see also [6]) we can show that this computes an Approximate Majority with high probability, which finishes the proof.[3]

---

[2] Indeed, this is inevitable with all constructions that we are aware of, since they are actually AC^0 circuits and it is known by a result of Håstad [5] that any AC^0 circuit of depth d for the Majority function must have size $\exp(\Omega(n^{1/(d-1)}))$.

[3] This is a slightly imprecise description of the construction as the final two levels of the circuit are actually defined somewhat differently.

2 Preliminaries

Throughout, n will be a growing parameter. We will consider Boolean functions on n variables, i.e. functions of the form $f : \{0,1\}^n \to \{0,1\}$. We will sometimes identify $\{0,1\}$ with the field $\mathbb{F}_2$ in the natural way and consider functions $f : \mathbb{F}_2^n \to \mathbb{F}_2$ instead.

Given a Boolean vector $y \in \{0,1\}^n$, we use $|y|_0$ and $|y|_1$ to denote the number of 0s and number of 1s respectively in y.

The Majority function on n variables, denoted MAJ_n is the Boolean function that maps inputs $x \in \{0,1\}^n$ to 1 if and only if $|x|_1 > n/2$.

▶ **Definition 2.** An (ε, n)-Approximate Majority is a function $f : \{0,1\}^n \to \{0,1\}$ such that $\Pr_{x \in \{0,1\}^n}[f(x) \neq \mathrm{Maj}_n(x)] \leq \varepsilon$.

As far as we know, the study of this class of functions was initiated by O'Donnell and Wimmer [11]. See also [1, 4].

We refer the reader to [2, 7] for standard definitions of Boolean circuits and formulas. We use AC^0 circuits (respectively formulas) to denote circuits (respectively formulas) of constant depth made up of AND, OR and NOT gates. Similarly, $\mathrm{AC}^0[\oplus]$ circuits (respectively formulas) will be circuits (respectively formulas) of constant depth made up of AND, OR, MOD_2 and NOT gates.

The size of a circuit will denote the number of gates in the circuit and the size of a formula will denote the number of its leaves which is within a constant multiplicative factor of the number of gates in the formula.[4]

3 Lower Bound

In this section, we show that any $\mathrm{AC}^0[\oplus]$ formulas of depth d computing a $(1/4, n)$-Approximate Majority must have size at least $\exp(\Omega(dn^{1/2(d-1)}))$ for all $d \leq O(\log n)$.

We work over the field $\mathbb{F}_2$ and identify it with $\{0,1\}$ in the natural way. The following concepts are standard in circuit complexity (see, e.g., Beigel's survey [3]).

▶ **Definition 3.** Fix any $\varepsilon \in [0,1]$. A polynomial $P \in \mathbb{F}_2[X_1, \ldots, X_n]$ is said to be an *ε-approximating polynomial* for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if

$$\Pr_{x \in \{0,1\}^n}[f(x) = P(x)] \geq 1 - \varepsilon.$$

We will use the following result of Smolensky [15] (see also Szegedy's PhD thesis [16]).

▶ **Lemma 4** (Smolensky [15])**.** *Let $\varepsilon \in (0, \frac{1}{2})$ be any fixed constant. Any $(\frac{1}{2} - \varepsilon)$-approximating polynomial for the Majority function on n variables must have degree $\Omega(\sqrt{n})$.*

▶ **Corollary 5.** *Let f be any $(1/4, n)$-Approximate Majority and $\varepsilon \in (0, 1/4)$ an arbitrary constant. Then any $(\frac{1}{4} - \varepsilon)$-approximating polynomial for f must have degree $\Omega(\sqrt{n})$.*

Proof. The proof is immediate from Lemma 4 and the triangle inequality. ◀

▶ **Definition 6.** An *ε-error probabilistic polynomial of degree D* for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a random variable $\mathbf{P}$ taking values from polynomials in $\mathbb{F}_2[X_1, \ldots, X_n]$ of degree at most D such that for all $x \in \{0,1\}^n$, we have $\Pr[\, f(x) = \mathbf{P}(x) \,] \geq 1 - \varepsilon$.

[4] We assume here without loss of generality that the formula does not contain a gate of fan-in 1 feeding into another.

▶ **Definition 7.** Let $D_\varepsilon(f)$ be the minimum degree of an ε-error probabilistic polynomial for f.

We will make use of the following two lemmas concerning $D_\varepsilon(\cdot)$.

▶ **Lemma 8** (Razborov [12])**.** *Let* OR_n *and* AND_n *be the OR and AND functions on* n *variables respectively. Then* $D_\varepsilon(\mathrm{OR}_n), D_\varepsilon(\mathrm{AND}_n) \leq \lceil \log(1/\varepsilon) \rceil$.

▶ **Lemma 9** (Kopparty and Srinivasan [10] (implicit in proof of Lemma 10))**.** *There is an absolute constant* c_1 *such that for any* $\varepsilon \in (0,1)$*,* $D_\varepsilon(f) \leq c_1 \cdot \lceil \log(1/\varepsilon) \rceil \cdot D_{1/8}(f)$ *for all Boolean functions* f*.*

We now state our main result, which shows that every $\mathrm{AC}^0[\oplus]$ formula of size s and depth $d+1$ admits a 1/8-error approximating polynomial of degree $O(\frac{1}{d}\log s)^d$.

▶ **Theorem 10.** *There is an absolute constant* c_2 *such that, if* f *is computed by an* $\mathrm{AC}^0[\oplus]$ *formula* F *of size* s *and depth* $d+1$*, then* $D_{1/8}(f) \leq 3(c_2(\frac{1}{d}\log(s)+1))^d$.

Proof. The proof is an induction on the depth d of the formula.

The base case $d = 0$ corresponds to the case when the formula is a single AND, OR or MOD_2 gate and we need to show that $D_{1/8}(f) \leq 3$. In the case that the formula is an AND or OR gate, this follows from Lemma 8. If the formula is a MOD_2 gate, this follows from the fact that the MOD_2 function is exactly a polynomial of degree 1.

Let $d \geq 1$. We assume that the formula F is the AND/OR/MOD_2 of sub-formulas $F_1, \ldots, F_m$ computing $f_1, \ldots, f_m$ where F_i has size s_i and depth $d+1$. So F has size $s = s_1 + \cdots + s_m$ and depth $d+2$. Assume that $D_{1/8}(f_i) \leq 3(c_2(\frac{1}{d}\log(s_i)+1))^d$ for all i. We must show that $D_{1/8}(f) \leq 3(c_2(\frac{1}{d+1}\log(s)+1))^{d+1}$.

By Lemma 9, each f_i has an $s_i/(16s)$-error probabilistic polynomial $\mathbf{P}_i$ of degree $c_1 \cdot \lceil \log(16s/s_i) \rceil \cdot D_{1/8}(f_i)$, which is at most

$$3c_1 \cdot 5(\log(s/s_i)+1) \cdot (c_2(\tfrac{1}{d}\log(s_i)+1))^d.$$

Then $(\mathbf{P}_1, \ldots, \mathbf{P}_m)$ jointly computes $(f_1, \ldots, f_m)$ with error 1/16 $(= \sum_{i=1}^m (s_i/(16s)))$.

By a reasoning identical to the base case, it follows that there exists a 1/16-error probabilistic polynomial $\mathbf{Q}$ of degree 4 for the output gate of the formula.

Then $\mathbf{Q}(\mathbf{P}_1, \ldots, \mathbf{P}_m)$ is a 1/8-error probabilistic polynomial for f of degree

$$60c_1 \cdot \max_i (\log(s/s_i)+1) \cdot (c_2(\tfrac{1}{d}\log(s_i)+1))^d.$$

So long as $c_2 \geq 20c_1$, it suffices to show that for all i,

$$(\log(s/s_i)+1) \cdot (\tfrac{1}{d}\log(s_i)+1)^d \leq (\tfrac{1}{d+1}\log(s)+1)^{d+1}.$$

Consider any i and let $a, b \geq 0$ such that $s_i = 2^a$ and $s = 2^{a+b}$. We must show

$$(b+1)\left(\frac{a}{d}+1\right)^d \leq \left(\frac{a+b}{d+1}+1\right)^{d+1}.$$

For fixed $a \geq 0$, as a polynomial in b, the function

$$p_{a,d}(b) := \left(\frac{a+b}{d+1}+1\right)^{d+1} - (b+1)\left(\frac{a}{d}+1\right)^d$$

is nonnegative over $b \geq 0$ with a unique root at $b = a/d$. This follows from

$$\frac{\partial}{\partial b} p_{a,d}(b) = \left(\frac{a+b}{d+1}+1\right)^d - \left(\frac{a}{d}+1\right)^d,$$

which is zero iff $b = a/d$; this value is a minimum of $p_{a,d}$ with $p_{a,d}(a/d) = 0$. ◀

▶ **Corollary 11.** *Fix any constant d and let $n \in \mathbb{N}$ be a growing parameter. Let f be any $(1/4, n)$-Approximate Majority. Then any $\mathrm{AC}^0[\oplus]$ formula of depth d computing f must have size $\exp(\Omega(dn^{1/2(d-1)}))$ for all $d \leq O(\log n)$, where asymptotic notation $O(\cdot)$ and $\Omega(\cdot)$ hide absolute constants (independent of d and n).*

Proof. Say that F is an $\mathrm{AC}^0[\oplus]$ formula of depth d and size s computing f. Then, by Lemma 9, we see that F has a $1/8$-error probabistic polynomial $\mathbf{P}$ of degree $D \leq O(O(\frac{1}{d}\log s + 1)^{d-1})$. In particular, by an averaging argument, there is some fixed polynomial $P \in \mathbb{F}_2[X_1, \ldots, X_n]$ of degree at most D such that P is a $1/8$-error approximating polynomial for f.

Corollary 5 implies that the degree of P must be $\Omega(\sqrt{n})$. Hence, we obtain $O(\frac{1}{d}\log s + 1)^{d-1} \geq \Omega(\sqrt{n})$. It follows that

$$s \geq \exp(\Omega(dn^{1/2(d-1)}) - O(d)).$$

Observe that $\Omega(dn^{1/2(d-1)})$ dominates $O(d)$ so long as $d \leq \varepsilon \log n$ for some absolute constant $\varepsilon > 0$ (depending on the constants in $\Omega(\cdot)$ and $O(\cdot)$). Hence, we get the claimed lower bound $s \geq \exp(\Omega(dn^{1/2(d-1)}))$ for all $d \leq \varepsilon \log n$. ◀

4 Upper Bound

In this section, we show that for any constant ε, there are (ε, n)-Approximate Majorities that can be computed by depth d AC^0 circuits of size $\exp(O(n^{1/2(d-1)}))$.

Let $\varepsilon_0 \in (0, 1)$ be a small enough constant so that the following inequalities hold for any $\beta \leq \varepsilon_0$

- $\exp(-\beta) \leq 1 - \beta\exp(-\beta)$,
- $1 - \beta \geq \exp(-\beta - \beta^2) \geq \exp(-2\beta)$.

(It suffices to take $\varepsilon_0 = 1/2$.)

We need the following technical lemma.

▶ **Lemma 12.** *Let A, s be positive reals, $M, n \in \mathbb{N}$, and $\gamma \in (\frac{1}{n}, \frac{1}{10})$ be such that $e^A \geq n^3$, $n \geq \frac{1}{\varepsilon_0}$, and $s \leq n$. Define $I_0(\gamma) := \{y \in \{0,1\}^M \mid |y|_1 \leq Me^{-A}(1-\gamma)\}$ and $I_1(\gamma) := \{y \in \{0,1\}^M \mid |y|_1 \geq Me^{-A}(1+\gamma)\}$. If we choose $S \subseteq [M]$ of size $t := \lceil e^A \cdot s \rceil$ by picking t random elements from M with replacement, then*

$$x \in I_0(\gamma) \Rightarrow \Pr_S[\bigvee_{j \in S} x_j = 0] \geq \exp(-s) \cdot \exp(s\gamma/2),$$

$$x \in I_1(\gamma) \Rightarrow \Pr_S[\bigvee_{j \in S} x_j = 0] \leq \exp(-s) \cdot \exp(-s\gamma).$$

Further, if $s\gamma \leq \varepsilon_0$, then the above probabilities can be lower bounded and upper bounded by $\exp(-s) \cdot (1 + s\gamma\exp(-s\gamma))$ and $\exp(-s) \cdot (1 - s\gamma\exp(-s\gamma))$ respectively.

A similar statement can be obtained above for the sets $J_1(\gamma) := \{y \in \{0,1\}^M \mid |y|_0 \leq Me^{-A}(1-\gamma)\}$ and $J_0(\gamma) := \{y \in \{0,1\}^M \mid |y|_0 \geq Me^{-A}(1+\gamma)\}$, with the event "$\bigvee_{j\in S} x_j = 0$" being replaced by the event "$\bigwedge_{j\in S} x_j = 1$".

Proof. We give the proof only for $I_0(\gamma)$ and $I_1(\gamma)$. The proof for $J_0(\gamma)$ and $J_1(\gamma)$ is similar.

Consider first the case that $x \in I_1(\gamma)$. In this case, we have the following computation.

$$\begin{aligned}\Pr_S[\bigvee_{j \in S} x_j = 0] &\leq \left(1 - \frac{1+\gamma}{e^A}\right)^{e^A \cdot s} \\ &\leq \exp(-(1+\gamma) \cdot s) \leq \exp(-s) \cdot \exp(-s\gamma). \qquad (1)\end{aligned}$$

The above implies the first upper bound on $\Pr_S[\bigvee_{j\in S} x_j = 0]$ from the lemma statement. When $s\gamma \le \varepsilon_0$, we further have $\exp(-s\gamma) \le 1 - s\gamma\exp(-s\gamma)$, which implies the second upper bound. This proves the lemma when $x \in I_1$.

Now consider the case that $x \in I_0(\gamma)$. We have

$$\begin{aligned}
\Pr_S[\bigvee_{j\in S} x_j = 0] &\ge \left(1 - \frac{1-\gamma}{e^A}\right)^{e^A\cdot s+1} \\
&\ge \exp\left((-\frac{1-\gamma}{e^A} - \frac{1}{e^{2A}})\cdot(e^A\cdot s+1)\right) \\
&= \exp\left(-s + s\gamma - \frac{s}{e^A} - \frac{1-\gamma}{e^A} - \frac{1}{e^{2A}}\right) \\
&\ge \exp\left(-s + s\gamma - \frac{2s}{e^A}\right) \\
&= \exp(-s)\cdot\exp(s\gamma(1 - \frac{2e^{-A}}{\gamma}))) \qquad (2)
\end{aligned}$$

where for the second inequality we have used the fact that since $e^{-A} \le \frac{1}{n^3} \le \varepsilon_0$, we have $1 - \frac{1-\gamma}{e^A} \ge \exp(-\frac{1-\gamma}{e^A} - \frac{1}{e^{2A}})$. Since $e^{-A} \le \frac{1}{n^3} \le \frac{1}{4n} \le \gamma/4$, we can lower bound the right hand side of (2) by $\exp(-s)\cdot\exp(s\gamma/2)$. Also, note that

$$\begin{aligned}
1 - \frac{2e^{-A}}{\gamma} &\ge 1 - \frac{2/n^3}{1/n} = 1 - \frac{2}{n^2} \\
&\ge \exp(-1/n) \ge \exp(-\gamma) \ge \exp(-s\gamma).
\end{aligned}$$

This implies that the RHS of (2) can also be lower bounded by $\exp(-s)\exp(s\gamma\exp(-s\gamma)) \ge \exp(-s)\cdot(1 + s\gamma\exp(-s\gamma))$, which implies the claim about $\Pr_S[\bigvee_{j\in S} x_j = 0]$ assuming that $x \in I_0(\gamma)$. ◀

We now prove the main result of this section.

▶ **Theorem 13.** *For any growing parameter $n \in \mathbb{N}$ and $2 \le d \le O(\frac{\log n}{\log\log n})$ and $\varepsilon > 0$, there is an (ε, n)-Approximate Majority f_n computable by a monotone AC^0 circuit with at most $\exp(O(n^{1/2(d-1)}\log(1/\varepsilon)/\varepsilon))$ many gates, where both $O(\cdot)$'s hide absolute constants (independent of d, ε).*

Proof. We assume throughout that ε is a small enough constant and that n is large enough for various inequalities to hold. We will actually construct a monotone circuit of depth d and size $\exp(O(n^{1/2(d-1)}\log(1/\varepsilon)/\varepsilon))$ computing a $(4\varepsilon, n)$-Approximate Majority, which also implies the theorem.

Fix parameters $A = \lfloor n^{1/2(d-1)} \rfloor$ and $M = \lceil e^{10A} \rceil$. We assume that $A \ge 10\log n$ (which holds as long as $d \le \frac{c\log n}{\log\log n}$ for an absolute constant $c > 0$) and that $\varepsilon \le \varepsilon_0$.

Define a sequence of real numbers $\gamma_0, \gamma_1, \ldots, \gamma_{d-2}$ as follows:

$$\begin{aligned}
\gamma_0 &= \frac{\varepsilon}{\sqrt{n}} \\
\gamma_i &= A\gamma_{i-1}\exp(-2A\gamma_{i-1}), \text{ for each } i \in [d-2].
\end{aligned}$$

It is clear that $\gamma_i \le A^i\gamma_0$ for each $i \in [d-2]$. As a result we also obtain

$$\begin{aligned}
\gamma_i &= A^i\gamma_0\exp(-2A(\gamma_0 + \gamma_1 + \cdots + \gamma_{i-1})) \\
&\ge A^i\gamma_0\exp(-2\gamma_0 A(1 + A + A^2 + \cdots + A^{i-1})) \ge A^i\gamma_0\exp(-3A^i\gamma_0). \qquad (3)
\end{aligned}$$

Let

$$Y_\varepsilon = \left\{ x \in \{0,1\}^n \;\middle|\; |x|_1 \geq \left(\frac{1}{2} + \frac{\varepsilon}{\sqrt{n}}\right) n \right\},$$
$$N_\varepsilon = \left\{ x \in \{0,1\}^n \;\middle|\; |x|_1 \leq \left(\frac{1}{2} - \frac{\varepsilon}{\sqrt{n}}\right) n \right\}.$$

The idea is to define a sequence of circuits $C_1, C_2, \ldots, C_{d-2}$ with n inputs and M outputs such that C_i has depth i and iM many (non-input) gates. Further, for odd i

$$\begin{aligned} x \in N_\varepsilon &\Rightarrow C_i(x) \in I_0(\gamma_i) \\ x \in Y_\varepsilon &\Rightarrow C_i(x) \in I_1(\gamma_i) \end{aligned} \tag{4}$$

and similarly for even i

$$\begin{aligned} x \in N_\varepsilon &\Rightarrow C_i(x) \in J_0(\gamma_i) \\ x \in Y_\varepsilon &\Rightarrow C_i(x) \in J_1(\gamma_i). \end{aligned} \tag{5}$$

After this is done, we will add on top a depth-2 circuit that will reject most inputs from $I_0(\gamma_{d-2})$ or $J_0(\gamma_{d-2})$ – depending on whether $d-2$ is odd or even respectively – and accept most inputs from $I_1(\gamma_{d-2})$ or $J_1(\gamma_{d-2})$.

We begin with the construction of $C_1, \ldots, C_{d-2}$ which is done by induction.

Construction of C_1. The base case of the induction is the construction of C_1, which is done as follows. We choose M i.i.d. random subsets $T_1, \ldots, T_M \subseteq [n]$ in the following way: for each $i \in [M]$, we sample A random elements of $[n]$ with replacement. Let $b_i^x = \bigwedge_{j \in T_i} x_j$.

If $x \in N_\varepsilon$, then the probability that $b_i^x = 1$ is given by

$$\Pr[b_i^x = 1] \leq \left(\frac{1}{2} - \gamma_0\right)^A \leq \frac{1}{2^A}(1 - 2\gamma_0)^A \leq \frac{1}{e^A}(1 - \gamma_0 A)$$

where the last inequality follows from the fact that $(1-z)^A \leq (1 - zA + \frac{A^2 z^2}{2})$.

Let $\delta = 1/n^3$. Note in particular that $2\delta/\gamma_0 A \leq \varepsilon_0$ for large enough n.

By a Chernoff bound, the probability that $\frac{1}{M}\sum_i b_i^x \geq \frac{1}{e^A}(1 - \gamma_0 A)(1+\delta)$ is bounded by $\exp(-\Omega(\delta^2 M/e^A)) \leq \exp(-\Omega(e^{9A}/n^6)) \leq \exp(-n)$, since $e^A \geq n^{10}$. Thus, with probability at least $1 - \exp(-n)$, we have

$$\begin{aligned} \frac{\sum_i b_i^x}{M} &\leq \frac{1}{e^A}(1 - \gamma_0 A)(1+\delta) \\ &\leq \frac{1}{e^A}(1 - \gamma_0 A + \delta) = \frac{1}{e^A}(1 - \gamma_0 A(1 - \frac{\delta}{\gamma_0 A})) \\ &\leq \frac{1}{e^A}(1 - \gamma_0 A \exp(-\frac{2\delta}{\gamma_0 A})) \leq \frac{1}{e^A}(1 - \gamma_0 A \exp(-\gamma_0 A)) \\ &\leq \frac{1}{e^A}(1 - \gamma_1). \end{aligned} \tag{6}$$

Above, we have used the fact that $(1 - \frac{\delta}{\gamma_0 A}) \geq \exp(\frac{-2\delta}{\gamma_0 A})$ since $\delta/\gamma_0 A \leq \varepsilon_0$ for large enough n, as noted above.

If $x \in Y_\varepsilon$, then the probability that $b_i^x = 1$ is given by

$$\begin{aligned}\Pr[b_i^x = 1] &\geq \left(\frac{1}{2} + \gamma_0\right)^A \\ &\geq \frac{1}{2^A}(1 + 2\gamma_0)^A \geq \frac{1}{e^A}(1 + \gamma_0 A) \\ &\geq \frac{1}{e^A}(1 + \gamma_0 A).\end{aligned}$$

As above, we can argue that the probability that $\frac{1}{M}\sum_i b_i^x \leq \frac{1}{e^A}(1 + \gamma_0 A)(1 - \delta)$ is at most $\exp(-n)$. Thus, with probability $1 - \exp(-n)$

$$\begin{aligned}\frac{\sum_i b_i^x}{M} &\geq \frac{1}{e^A}(1 + \gamma_0 A)(1 - \delta) \\ &\geq \frac{1}{e^A}(1 + \gamma_0 A - 2\delta) = \frac{1}{e^A}(1 + \gamma_0 A(1 - \frac{2\delta}{\gamma_0 A})) \\ &\geq \frac{1}{e^A}(1 + \gamma_0 A \exp(-\frac{4\delta}{\gamma_0 A})) \geq \frac{1}{e^A}(1 + \gamma_0 A \exp(-\gamma_0 A)) \\ &\geq \frac{1}{e^A}(1 + \gamma_1). \qquad (7)\end{aligned}$$

Thus, by a union bound over x, we can fix a choice of $T_1, \ldots, T_M$ so that (6) holds for all $x \in N_\varepsilon$ and (7) holds for all $x \in Y_\varepsilon$. Hence, (4) holds for $i = 1$ as required. This concludes the construction of C_1, which just outputs the values of $\bigwedge_{j \in T_i} x_j$ for each i.

Construction of C_{i+1}. For the inductive case, we proceed as follows. We assume that i is odd (the case that i is even is similar). So by the inductive hypothesis, we know that (4) holds and hence that $C_i(x) \in I_0(\gamma_i)$ or $I_1(\gamma_i)$ depending on whether $x \in N_\varepsilon$ or Y_ε. Let $\gamma := \gamma_i$. Let the output gates of C_i be $g_1, \ldots, g_M$.

We choose $T_1, \ldots, T_M \subseteq [M]$ randomly as in the statement of Lemma 12 with $s = A$. Note that the chosen parameters satisfy all the hypotheses of Lemma 12. Further we also have $s\gamma \leq A \cdot A^i\gamma_0 \leq A^{d-1} \cdot \frac{\varepsilon}{\sqrt{n}} \leq \varepsilon_0$.

The random circuit C' is defined to be the circuit obtained by adding M OR gates to C_i such that the jth OR gate computes $\bigvee_{k \in T_j} g_k$. Let b_j^x be the output of the jth OR gate on $C_i(x)$.

By Lemma 12, we have

$$\begin{aligned}x \in N_\varepsilon &\Rightarrow \Pr_S[b_j^x = 0] \geq \exp(-A) \cdot (1 + A\gamma\exp(-A\gamma)) \\ x \in Y_\varepsilon &\Rightarrow \Pr_S[b_j^x = 0] \leq \exp(-A) \cdot (1 - A\gamma\exp(-A\gamma)) \qquad (8)\end{aligned}$$

Let $\delta = \frac{1}{n^3}$. Note that $A\gamma \in [\frac{1}{\sqrt{n}}, \frac{1}{n^{1/2(d-1)}}]$ and hence for large enough n, $\frac{2\delta}{A\gamma\exp(-A\gamma)} \leq \varepsilon_0$.

Assume $x \in N_\varepsilon$. In this case, the Chernoff bound implies that the probability that $\sum_{j \in [M]} b_j^x \leq M\exp(-A) \cdot (1 + A\gamma\exp(-A\gamma))(1 - \delta)$ is at most $\exp(-\Omega(\delta^2 M/e^A)) \leq \exp(-n)$.

When this event does not occur, we have

$$\begin{aligned}\frac{\sum_i b_i^x}{M} &\geq \frac{1}{e^A}(1 + A\gamma\exp(-A\gamma))(1-\delta)\\ &\geq \frac{1}{e^A}(1 + A\gamma\exp(-A\gamma) - 2\delta) = \frac{1}{e^A}(1 + A\gamma\exp(-A\gamma)(1 - \frac{2\delta}{A\gamma\exp(-A\gamma)}))\\ &\geq \frac{1}{e^A}(1 + A\gamma\exp(-A\gamma)\cdot\exp(-\frac{4\delta}{A\gamma\exp(-A\gamma)}))\\ &\geq \frac{1}{e^A}(1 + A\gamma\exp(-A\gamma)\cdot\exp(-A\gamma))\\ &\geq \frac{1}{e^A}(1 + A\gamma\exp(-2A\gamma)) \geq \frac{1}{e^A}(1+\gamma_{i+1}). \qquad (9)\end{aligned}$$

We have used above that for large enough n, $\frac{2\delta}{A\gamma\exp(-A\gamma)} \leq \varepsilon_0$ and hence $1 - \frac{2\delta}{A\gamma\exp(-A\gamma)} \geq \exp(\frac{-4\delta}{A\gamma\exp(-A\gamma)})$.

Similarly when $x \in Y_\varepsilon$, the Chernoff bound tells us that the probability that $\sum_{j\in[M]} b_j^x \geq M\exp(-A)\cdot(1 - A\gamma\exp(-A\gamma))(1+\delta)$ is at most $\exp(-n)$. In this case, we get

$$\begin{aligned}\frac{\sum_i b_i^x}{M} &\leq \frac{1}{e^A}(1 - A\gamma\exp(-A\gamma))(1+\delta)\\ &\leq \frac{1}{e^A}(1 - A\gamma\exp(-A\gamma) + \delta) = \frac{1}{e^A}(1 - A\gamma\exp(-A\gamma)(1 - \frac{\delta}{A\gamma\exp(-A\gamma)}))\\ &\leq \frac{1}{e^A}(1 - A\gamma\exp(-A\gamma)\cdot\exp(-\frac{2\delta}{A\gamma\exp(-A\gamma)}))\\ &\leq \frac{1}{e^A}(1 - A\gamma\exp(-A\gamma)\cdot\exp(-A\gamma))\\ &= \frac{1}{e^A}(1 - A\gamma\exp(-2A\gamma)) \geq \frac{1}{e^A}(1-\gamma_{i+1}). \qquad (10)\end{aligned}$$

By a union bound, we can fix $T_1, \ldots, T_M$ so that (9) and (10) are true for all $x \in N_\varepsilon$ and $x \in Y_\varepsilon$ respectively. This gives us the circuit C_{i+1} which satisfies all the required properties.

The top two levels of the circuit. At the end of the above procedure we have a circuit C_{d-2} of depth $d-2$ and at most $(d-2)M$ gates that satisfies one of (4) or (5) depending on whether $d-2$ is odd or even respectively. We assume that $d-2$ is even (the other case is similar).

Define $\gamma := \gamma_{d-2}$. Recall from (3) that $\gamma \geq A^{d-2}\gamma_0\exp(-3A^{d-2}\gamma_0) \geq A^{d-2}\gamma_0/2$.

Let $M' = \lceil\exp(\frac{10A\log(1/\varepsilon)}{\varepsilon} + 10A)\rceil$. We choose M' many subsets $T_1, \ldots, T_{M'} \subseteq [M]$ i.i.d. so that each T_j is picked as in Lemma 12 with $s = 10A\log(1/\varepsilon)/\varepsilon$. Note that

$$s\gamma \geq s\frac{A^{d-2}\gamma_0}{2} = \frac{10A\log(1/\varepsilon)}{\varepsilon}\cdot\frac{A^{d-2}}{2}\cdot\frac{\varepsilon}{\sqrt{n}} \geq 5\log(1/\varepsilon).$$

Say $g_1, \ldots, g_M$ are the output gates of C_{d-2}. We define the random circuit C' (with n inputs and M' outputs) to be the circuit obtained by adding M' AND gates such that the jth AND gate computes $\bigwedge_{k\in T_j} g_k$. Let b_j^x be the output of the jth AND gate on $C_{d-2}(x)$.

By Lemma 12, we have

$$\begin{aligned}x \in N_\varepsilon &\Rightarrow \Pr_S[b_j^x = 1] \leq \exp(-s)\cdot\exp(-s\gamma) \leq \varepsilon^2\cdot\exp(-s)\\ x \in Y_\varepsilon &\Rightarrow \Pr_S[b_j^x = 1] \geq \exp(-s)\cdot\exp(s\gamma/2) \geq \frac{\exp(-s)}{\varepsilon^2}. \qquad (11)\end{aligned}$$

Say $x \in N_\varepsilon$. By a Chernoff bound, the probability that $\sum_j b_j^x \geq 2\varepsilon^2 M' \exp(-s)$ is at most $\exp(-\Omega(\varepsilon^2 M' \exp(-s))) \leq \exp(-\Omega(\varepsilon^2 e^{10A})) \leq \exp(-n)$. Similarly, when $x \in Y_\varepsilon$, the probability that $\sum_j b_j^x \leq \frac{M' \exp(-s)}{2\varepsilon^2}$ is also bounded by $\exp(-n)$. By a union bound, we can fix a $T_1, \ldots, T_{M'}$ to get a circuit C_{d-1} such that

$$x \in N_\varepsilon \Rightarrow |C_{d-1}(x)|_1 \leq 2\varepsilon^2 \exp(-s) M'$$
$$x \in Y_\varepsilon \Rightarrow |C_{d-1}(x)|_1 \geq \frac{1}{2\varepsilon^2} \exp(-s) M'. \tag{12}$$

This gives us the depth $d-1$ circuit C_{d-1}. Note that C_{d-1} has $M' + O(dM) = O(M')$ gates.

To get the depth d circuit, we choose a random subset $T \subseteq [M']$ by sampling exactly $\lceil \exp(s) \rceil$ many elements of $[M']$ with replacement. We construct a random depth-d circuit C'_d by taking the OR of the the output gates of C_{d-1} indexed by the subset T.

From (12) it follows that

$$x \in N_\varepsilon \Rightarrow \Pr_T[C'_d(x) = 1] \leq |T| \cdot 2\varepsilon^2 \exp(-s) \leq 4\varepsilon^2 < \varepsilon$$
$$x \in Y_\varepsilon \Rightarrow \Pr_T[C'_d(x) = 0] \leq \left(1 - \frac{\exp(-s)}{2\varepsilon^2}\right)^{\exp(s)} \leq \exp(-1/2\varepsilon^2) < \varepsilon.$$

The final inequalities in each case above hold as long as ε is a small enough constant.

It follows from the above that there is a choice for T such that C'_d makes an error – i.e. $C'_d(x) = 1$ for $x \in N_\varepsilon$ or $C'_d(x) = 0$ for $x \in Y_\varepsilon$ – on at most a 2ε fraction of inputs from $N_\varepsilon \cup Y_\varepsilon$. We fix such a choice for T and the corresponding circuit C.

We have

$$\begin{aligned}\Pr_{x \in \{0,1\}^n}[C(x) \neq \mathrm{Maj}_n(x)] &\leq \Pr_{x \in Y_\varepsilon \cup N_\varepsilon}[C(x) \neq \mathrm{Maj}_n(x)] + \Pr_{x \in \{0,1\}^n}[x \notin Y_\varepsilon \cup N_\varepsilon] \\ &\leq 2\varepsilon + \Pr_{x \in \{0,1\}^n}[x \notin Y_\varepsilon \cup N_\varepsilon].\end{aligned}$$

Finally by Stirling's approximation we get

$$\Pr_{x \in \{0,1\}^n}[x \notin Y_\varepsilon \cup N_\varepsilon] = \frac{1}{2^n} \sum_{m \in [\frac{n}{2} - \varepsilon\sqrt{n}, \frac{n}{2} + \varepsilon\sqrt{n}]} \binom{n}{m} \leq \frac{1}{2^n} \sum_{m \in [\frac{n}{2} - \varepsilon\sqrt{n}, \frac{n}{2} + \varepsilon\sqrt{n}]} \binom{n}{n/2} \leq 2\varepsilon.$$

Hence we see that the circuit C computes a $(4\varepsilon, n)$-Approximate Majority, which proves Theorem 13.

The circuit has depth d and size $O(M') = \exp(O(n^{1/2(d-1)} \log(1/\varepsilon)/\varepsilon))$. ◀

5 Conclusion

Our main results extend straightforwardly to $\mathrm{AC}^0[\mathrm{MOD}_p]$ for any fixed prime p. The proofs are exactly the same except for the fact that the approximating polynomials of degree $O(\frac{1}{d} \log s)^{d-1}$ from Section 3 are constructed over $\mathbb{F}_p$.

Using the fact [14] that any (1/4)-approximating polynomial over $\mathbb{F}_p$ (p odd) for the Parity function on n variables must have degree $\Omega(\sqrt{n})$, we see that any polynomial-sized $\mathrm{AC}^0[\mathrm{MOD}_p]$ formula computing the Parity function on n variables must have depth $\Omega(\log n)$. This strengthens a result of Rossman [13] which gives this statement for AC^0 formulas.

Acknowledgements. We thank Rahul Santhanam for valuable discussions. We also thank the organizers of the 2016 Complexity Semester at St. Petersburg, where this collaboration began.

References

1 Kazuyuki Amano. Bounds on the size of small depth circuits for approximating majority. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 59–70, 2009. doi:10.1007/978-3-642-02927-1_7.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

3 Richard Beigel. The polynomial method in circuit complexity. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 82–95, 1993. doi:10.1109/SCT.1993.336538.

4 Eric Blais and Li-Yang Tan. Approximating boolean functions with depth-2 circuits. *SIAM J. Comput.*, 44(6):1583–1600, 2015. doi:10.1137/14097402X.

5 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986. doi:10.1145/12130.12132.

6 Shlomo Hoory, Avner Magen, and Toniann Pitassi. Monotone circuits for the majority function. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 410–425. Springer, 2006.

7 Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.

8 Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990.

9 Maria M. Klawe, Wolfgang J. Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth (preliminary version). In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 – May 2, 1984, Washington, DC, USA*, pages 480–487, 1984. doi:10.1145/800057.808717.

10 Swastik Kopparty and Srikanth Srinivasan. Certifying polynomials for $AC^0[\oplus]$ circuits, with applications. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 18. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

11 Ryan O'Donnell and Karl Wimmer. Approximation by DNF: examples and counterexamples. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 195–206, 2007. doi:10.1007/978-3-540-73420-8_19.

12 Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987.

13 Benjamin Rossman. The average sensitivity of bounded-depth formulas. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 424–430. IEEE, 2015.

14 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82, 1987. doi:10.1145/28395.28404.

15 Roman Smolensky. On representations by low-degree polynomials. In *FOCS*, pages 130–138, 1993. doi:10.1109/SFCS.1993.366874.

16 Mario Szegedy. *Algebraic Methods in Lower Bounds for Computational Models with Limited Communication*. PhD thesis, University of Chicago, 1989.

17 Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984. doi:10.1016/0196-6774(84)90016-6.

Sensitivity Conjecture and Log-Rank Conjecture for Functions with Small Alternating Numbers*

Chengyu Lin[†1] **and Shengyu Zhang**[2]

1 **Columbia University, New York, NY, USA**
chengyu@cs.columbia.edu
2 **The Chinese University of Hong Kong, Hong Kong, China**
syzhang@cse.cuhk.edu.hk

Abstract

The Sensitivity Conjecture and the Log-rank Conjecture are among the most important and challenging problems in concrete complexity. Incidentally, the Sensitivity Conjecture is known to hold for monotone functions, and so is the Log-rank Conjecture for $f(x \wedge y)$ and $f(x \oplus y)$ with monotone functions f, where $\wedge$ and $\oplus$ are bit-wise AND and XOR, respectively. In this paper, we extend these results to functions f which alternate values for a relatively small number of times on any monotone path from 0^n to 1^n. These deepen our understandings of the two conjectures, and contribute to the recent line of research on functions with small alternating numbers.

1998 ACM Subject Classification F.1.3 [Complexity Measures and Classes] Relations among Complexity Measures

Keywords and phrases Analysis of Boolean functions, Sensitivity Conjecture, Log-rank Conjecture, Alternating Number

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.51

1 Introduction

A central topic in Boolean function complexity theory is relations among different combinatorial and computational measures [29]. For Boolean functions, there is a large family of complexity measures such as block sensitivity, certificate complexity, decision tree complexity (including its randomized and quantum versions), degree (including its approximate version), etc, that are all polynomially related [13]. One outlier[1] is sensitivity, which a priori could be exponentially smaller than the ones in that family. The famous Sensitivity Conjecture raised by Nisan and Szegedy [48] says that sensitivity is also polynomially related to the block sensitivity and others in the family. Despite a lot of efforts, the best upper bound we know is still exponential: $\mathsf{bs}(f) \le \mathsf{C}(f) \le \left(\frac{8}{9} + o(1)\right) \mathsf{s}(f) 2^{\mathsf{s}(f)-1}$ from [25], improving upon previous work [54, 2, 3]. See a recent survey [24] about this conjecture and how it has resisted many serious attacks.

Communication complexity quantifies the minimum amount of communication required for computing functions whose inputs are distributed among two or more parties [31]. In

* This work was supported by Research Grants Council of the Hong Kong S.A.R. (Project no. CUHK14239416). The first author was also supported by NSF grants #CNS-1445424 and #CCF-1423306.

† Most of this work was done when Chengyu Lin was in Chinese University of Hong Kong.

1 There are complexity measures, such as $\mathbb{F}_2$-degree, polynomial threshold degree, total influence, Boolean circuit depth, CNF/DNF size, that are *known* not to belong to the polynomially equivalent class. But the position of sensitivity is elusive.

the standard bipartite setting, the function F has two inputs x and y, with x given to Alice and y to Bob. The minimum number of bits needed to be exchanged to compute $F(x,y)$ for all inputs (x,y) is the communication complexity $\mathsf{CC}(F)$. It has long been known [41] that the logarithm of the rank of communication matrix $M_F \stackrel{\text{def}}{=} [F(x,y)]_{x,y}$ is a lower bound of $\mathsf{CC}(F)$. Perhaps the most prominent and long-standing open question about communication complexity is the Log-rank Conjecture proposed by Lovász and Saks [37], which asserts that $\mathsf{CC}(F)$ of any Boolean function F is also upper bounded by a polynomial of $\log \mathsf{rank}(M_F)$. The conjecture has equivalent forms related to chromatic number conjecture in graph theory [37], nonnegative rank [36], Boolean roots of polynomials over real numbers [61], quantum sampling complexities [4, 65], etc. Despite a lot of efforts devoted to the conjecture in the past decades, the best upper bound is $\mathsf{CC}(F) = O\big(\sqrt{\mathsf{rank}(M_F)}\log\left(\mathsf{rank}(M_F)\right)\big)$ by Lovett [39], which is still exponentially far from the target.

While these two conjectures are notoriously challenging in their full generality, special classes of functions have been investigated. In particular, the Sensitivity Conjecture is confirmed to hold for monotone functions, as the sensitivity coincides with block sensitivity and certificate complexity for those functions [47]. The Log-rank Conjecture is not known to be true for monotone functions, but it holds for monotone functions on two bit-wise compositions between x and y. More specifically, two classes of bit-wise composed functions have drawn substantial attention. The first class contains AND functions $F = f \circ \wedge$, defined by $F(x,y) = f(x \wedge y)$, where $\wedge$ is the bit-wise AND of $x, y \in \{0,1\}^n$. Taking the outer function f to be the n-bit OR, we get Disjointness, the function that has had a significant impact on both communication complexity theory itself [53] and applications to many other areas such as streaming, data structures, circuit complexity, proof complexity, game theory and quantum computation [15]. The AND functions also contain other well known functions such as Inner Product, AND-OR trees [28, 33, 27, 19], and functions exhibiting gaps between communication complexity and log-rank [49]. The second class is XOR functions $F = f \circ \oplus$, defined by $F(x,y) = f(x \oplus y)$, where $\oplus$ is the bit-wise XOR function. This class includes Equality [62, 46, 1, 7, 11] and Hamming Distance [63, 16, 26, 34, 35] as special cases.

Both AND and XOR functions have recently drawn much attention [38, 12, 67, 32, 42, 55, 35, 59, 66, 50, 64], partly because their communication matrix rank has intimate connections to the polynomial representations of the outer function f. Specifically, the rank of $M_{f\circ\wedge}$ is exactly the Möbius sparsity[2] $\mathtt{mono}(f)$, the number of nonzero coefficients $\alpha(S)$ in the multilinear polynomial representation $f(x) = \sum_{S\subseteq[n]} \alpha(S) \prod_{i\in S} x_i$ for $f : \{0,1\}^n \to \{0,1\}$ [12]. And the rank of $M_{f\circ\oplus}$ is exactly the Fourier sparsity $\|\hat{f}\|_0$, the number of nonzero Fourier coefficients $\hat{f}(S)$ in the multilinear polynomial representation $f(x) = \sum_{S\subseteq[n]} \hat{f}(S) \prod_{i\in S} x_i$ for $f : \{+1,-1\}^n \to \{0,1\}$.

It is known that the Log-rank Conjecture holds for these two classes of functions when the outer function f is monotone [38, 42], and this work aims to extend these as well as the sensitivity result on monotone functions, to functions that are *close to* being monotone. One needs to be careful about the distance measure here, since the widely-used (e.g. in property testing and computational learning) normalized Hamming distance $\mathsf{dist}(f,g) = \Pr_{x\in\{0,1\}^n}[f(x) \neq g(x)]$ does not meet our requirement. Indeed, if we flip the value $f(x)$ at just one input x, then this changes f by an exponentially small amount measured by dist, but the sensitivity would change from a small $\mathsf{s}(f)$ to a large $n - \mathsf{s}(f)$. Similarly, the Fourier sparsity is also very sensitive to local changes ($\|\hat{f}\|_0$ to $2^n - \|\hat{f}\|_0$), and so is Möbius sparsity

[2] Named after the Möbius transform from f to α.

if we flip the value at 0^n.

One robust distance measure to monotone functions, which has recently drawn an increasingly amount of attention, is the alternating number, defined as follows. View the Boolean hypercube $\{0,1\}^n$ as a lattice with the partial order $x \preceq y$ if $x_i \leq y_i$ for all i. A path $x^{(1)} \to \cdots \to x^{(k)}$ on $\{0,1\}^n$ is monotone if $x^{(i)} \prec x^{(i+1)}$ for all i. The alternating number of a function f on $\{0,1\}^n$ is the maximum number of i's with $f(x^{(i)}) \neq f(x^{(i+1)})$, on any monotone path $x^{(0)} \to \cdots \to x^{(n)}$ from 0^n to 1^n. It is clear that constant functions have alternating number 0, and monotone functions have alternating number 1. For general functions f, we have $\mathsf{alt}(f) \leq n$, thus $\mathsf{alt}(f)$ is a sub-linear complexity measure. The smaller $\mathsf{alt}(f)$ is, the closer it is to monotone functions. Studies of the alternating number dated back to [40], in which Markov showed that the inversion complexity, the minimum number of negation gates needed in any Boolean circuit computing f, is exactly $\lceil \log_2(\mathsf{alt}(f)+1) \rceil$. Late work investigated the inversion complexity/alternating number over computational models such as constant-depth circuit [52], bounded-depth circuit [56], Boolean formula [43], and non-deterministic circuit [44]. It has been recently shown that small alternating number can be exploited in learning Boolean circuits [10]. Also there are some studies in cryptography considering the effect of negation gates [23].

In this paper, we study the Sensitivity and Log-rank Conjectures for functions whose alternating numbers are small, compared to sensitivity, Möbius sparsity and Fourier sparsity. First, the following theorem shows that the Sensitivity Conjecture holds for f with $\mathsf{alt}(f) = \mathrm{poly}(\mathsf{s}(f))$.

▶ **Theorem 1.** *For any function $f : \{0,1\}^n \to \{0,1\}$, it holds that*

$$\mathsf{bs}(f) = O(\mathsf{alt}(f)^2 \cdot \mathsf{s}(f)).$$

Note that if a function is non-degenerate in the sense that it depends on all n variables, then the sensitivity is at least $\Omega(\log n)$ [54], therefore the above theorem also confirms the Sensitivity Conjecture for non-degenerate functions f with $\mathsf{alt}(f) = \mathrm{poly}\log n$.

The next two theorems confirmed the Log-rank Conjecture for $f \circ \oplus$ with $\mathsf{alt}(f) = \mathrm{poly}\log(\|\hat{f}\|_0)$, and for $f \circ \wedge$ with $\mathsf{alt}(f) = O(1)$.

▶ **Theorem 2.** *For any function $f : \{0,1\}^n \to \{0,1\}$, it holds that*

$$\mathsf{CC}(f \circ \oplus) \leq 2 \cdot \mathsf{alt}(f) \cdot \log^2 \mathsf{rank}(M_{f\circ\oplus}).$$

▶ **Theorem 3.** *For any function $f : \{0,1\}^n \to \{0,1\}$, it holds that*

$$\mathsf{CC}(f \circ \wedge) \leq O(\log^{\mathsf{alt}(f)+1} \mathsf{rank}(M_{f\circ\wedge})).$$

In the last theorem, the dependence on $\mathsf{alt}(f)$ can be slightly improved (by a factor of 2) if a factor of $\log n$ is tolerated in the communication cost.

Related work

The Sensitivity Conjecture has many equivalent forms, summarized in the survey [24]. Also see the recent paper [18] which tries to solve this conjecture using a communication game approach. At the other end of the spectrum, [51, 5] seek the largest possible separation between sensitivity and block sensitivity, and [9] got a super-quadratic separation between sensitivity and query complexity. Apart from monotone functions [47], the Sensitivity Conjecture has also been confirmed on graph properties [60], cyclically-invariant function [14],

read-once functions [45], functions admitting the Normalized Block property [57] and several cases of read-k formulas [8]. Other than the conjecture itself, some recent work [6, 21, 22] discussed combinatorial and computational structures of low-sensitivity functions.

For the Log-rank Conjecture, apart from the equivalent forms mentioned earlier, some seemingly weaker formulations in terms of the largest monochromatic rectangle size [49], randomized communication complexity and information cost [17] are actually equivalent to the original conjecture. For lower bounds, the best one had been $\mathsf{CC}(F) = \Omega\left((\log \mathsf{rank}(M_F))^{\log_3 6}\right)$ (attributed to Kushilevitz in [49]), achieved by an AND function, until the recent result of $\mathsf{CC}(F) = \tilde{\Omega}\left(\log^2 \mathsf{rank}(M_F)\right)$ [20]. For XOR functions $f \circ \oplus$, the Log-rank Conjecture is confirmed when f is symmetric [67], monotone [42], linear threshold functions (LTFs) [42], AC^0 functions [30], has low $\mathbb{F}_2$-degree [59] or small spectral norm [59]. For AND functions $f \circ \wedge$, it seems that the conjecture is only confirmed on monotone functions [38].

2 Preliminaries

2.1 n-bit (Boolean) functions

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. The all-0 n-bit string is denoted by 0^n and the all-1 n-bit string is denoted by 1^n.

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, its $\mathbb{F}_2$-degree is the degree of f viewed as a polynomial over $\mathbb{F}_2$. Such functions f can be also viewed as polynomials over $\mathbb{R}$: $f(x) = \sum_{S\subseteq[n]} \alpha(S)x^S$, where $x^S = \prod_{i\in S} x_i$. If we represent the domain by $\{+1,-1\}^n$, then the polynomial (still over $\mathbb{R}$) changes to $f(x) = \sum_{S\subseteq[n]} \hat{f}(S)x^S$, usually called Fourier expansion of f. The coefficients $\alpha(S)$ and $\hat{f}(S)$ in the two $\mathbb{R}$-polynomial representations capture many important combinatorial properties of f. We denote by $\mathtt{mono}(f)$ the Möbius sparsity, the number of non-zero coefficients $\alpha(S)$, and by $\|\hat{f}\|_0$ the Fourier sparsity, the number of non-zero coefficients $\hat{f}(S)$. Some basic facts used in this paper are listed as follows.

▶ **Fact 4.** *For any $f : \{0,1\}^n \to \{0,1\}$, $\deg_2(f) = n$ if and only if $|f^{-1}(1)|$ is odd.*

▶ **Fact 5.** *For any $f : \{0,1\}^n \to \{0,1\}$, $\deg_2(f) \le \log \|\hat{f}\|_0$.*

For any input $x \in \{0,1\}^n$ and $i \in [n]$, let x^i be the input obtained from x by flipping the value of x_i. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and an input x, if $f(x) \neq f(x^i)$, then we say that x is *sensitive to coordinate* i, and i is a *sensitive coordinate* of x. We can also define these for blocks. For a set $B \subseteq [n]$, let x^B be the input obtained from x by flipping x_i for all $i \in B$. Similarly, if $f(x) \neq f(x^B)$, then we say that x is *sensitive to block* B, and B is a *sensitive block* of x. The *sensitivity* $\mathsf{s}(f,x)$ *of function* f *on input* x is the number of sensitive coordinates i of x: $\mathsf{s}(f,x) = |\{i \in [n] : f(x) \neq f(x^i)\}|$, and the *sensitivity of function* f is $\mathsf{s}(f) = \max_x \mathsf{s}(f,x)$. It is easily seen that the n-bit AND and OR functions both have sensitivity n. The *block sensitivity* $\mathsf{bs}(f,x)$ *of function* f *on input* x is the maximal number of disjoint sensitive blocks of x, and the *block sensitivity of function* f is $\mathsf{bs}(f) = \max_x \mathsf{bs}(f,x)$. Note that there are always $\mathsf{bs}(f,x)$ many disjoint *minimal* sensitive blocks, in the sense that any $B \subsetneq B_i$ is not a sensitive block of x.

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and an input $x \in \{0,1\}^n$, the *certificate complexity* $\mathsf{C}(f,x)$ *of function* f *on input* x is the minimal number of variables restricting the value of which fixes the function to a constant. The *certificate complexity* of f is $\mathsf{C}(f) = \max_x \mathsf{C}(f,x)$, and the *minimal certificate complexity* of f is $\mathsf{C}_{\min}(f) = \min_x \mathsf{C}(f,x)$. The *decision tree complexity* $\mathsf{DT}(f)$ of function f is the minimum depth of any decision tree that computes f.

A *subfunction* or a *restriction* of a function f on $\{0,1\}^n$ is obtained from f by restricting the values of some variables. Sometimes we say to restrict f to *above* an input d, or to take the subfunction f' over $\{x : x \succeq d\}$, then we mean to restrict variables x_i to be 1 whenever $d_i = 1$. Similarly, we say to restrict f to *under* an input u, or take the subfunction f' over $\{x : x \preceq u\}$, meaning to restrict x_i to be 0 whenever $u_i = 0$.

Let $\mathcal{F}_n$ be the set of all the real-valued functions on $\{0,1\}^n$. A complexity measure $M : \cup_{n=1}^{\infty}\mathcal{F}_n \to \mathbb{R}$ is *downward non-increasing* if $M(f') \leq M(f)$ for all subfunction f' of f. That is, restricting variables does not increase the measure M. It is easily seen that the $\mathbb{F}_2$-degree, the alternating number, the decision tree complexity, the sensitivity, the block sensitivity, the certificate complexity, the Fourier sparsity, are all downward non-increasing. When M is not downward non-increasing, it makes sense to define the *closure* by $M^{\text{clo}}(f) = \max_{f'} M(f')$ where the maximum is taken over all subfunctions f' of f. In particular, $\mathsf{C}^{\text{clo}}_{\min}(f) = \max_{f'} \mathsf{C}_{\min}(f')$. The next theorem relates decision tree complexity to $\mathsf{C}^{\text{clo}}_{\min}$.

▶ **Theorem 6** ([59]). *For any $f : \{0,1\}^n \to \{0,1\}$, it holds that $\mathsf{DT}(f) \leq \mathsf{C}^{\text{clo}}_{\min}(f) \deg_2(f)$.*

(The original theorem proved was actually $\mathsf{PDT}(f) \leq \mathsf{C}^{\text{clo}}_{\oplus,\min}(f) \deg_2(f)$, where $\mathsf{PDT}(f)$ is the parity decision tree complexity and $\mathsf{C}^{\text{clo}}_{\oplus,\min}(f)$ is the parity minimum certificate complexity. But as observed by [58], the same argument applies to standard decision tree as well.)

For general Boolean functions f, we have $\mathsf{s}(f) \leq \mathsf{bs}(f) \leq \mathsf{C}(f)$. But when f is monotone, equalities are achieved.

▶ **Fact 7.** *If $f : \{0,1\}^n \to \{0,1\}$ is monotone, then $\mathsf{s}(f) = \mathsf{bs}(f) = \mathsf{C}(f)$.*

▶ **Fact 8** ([42]). *If $f : \{0,1\}^n \to \{0,1\}$ is monotone, then $\mathsf{s}(f) \leq \deg_2(f)$.*

One can associate a partial order $\preceq$ to the Boolean hypercube $\{0,1\}^n$: $x \preceq y$ if $x_i \leq y_i$ for all i. We also write $y \succeq x$ when $x \preceq y$. If $x \preceq y$ but $x \neq y$, then we write $x \prec y$ and $y \succ x$. A path $x^{(1)} \to \cdots \to x^{(k)}$ on $\{0,1\}^n$ is *monotone* if $x^{(i)} \prec x^{(i+1)}$ for all i.

▶ **Definition 9.** For any function on $\{0,1\}^n$, the *alternating number of a path* $x^{(1)} \to \cdots \to x^{(k)}$ is the number of $i \in \{1,2,...,k-1\}$ with $f(x^{(i)}) \neq f(x^{(i+1)})$. The alternating number $\mathsf{alt}(f,x)$ of input $x \in \{0,1\}^n$ is the maximum alternating number of any monotone path from 0^n to x, and the *alternating number of a function* f is $\mathsf{alt}(f) = \mathsf{alt}(f, 1^n)$. Equivalently, one can also define $\mathsf{alt}(f)$ to be the largest k such that there exists a list $\{x^{(1)}, x^{(2)}, \ldots, x^{(k+1)}\}$ with $x^{(i)} \preceq x^{(i+1)}$ and $f(x^{(i)}) \neq f(x^{(i+1)})$, for all $i \in [k]$.

A function $f : \{0,1\}^n \to \mathbb{R}$ is *monotone* if $f(x) \leq f(y)$, $\forall x \preceq y$. A function $f : \{0,1\}^n \to \mathbb{R}$ is *anti-monotone* if $f(x) \leq f(y)$, $\forall x \succeq y$. It is not hard to see that $\mathsf{alt}(f) = 0$ iff f is constant, and $\mathsf{alt}(f) = 1$ iff f is monotone or anti-monotone.

▶ **Definition 10.** For a function f on $\{0,1\}^n$, an input $u \in \{0,1\}^n - \{1^n\}$ is called a *max term* if $f(u) \neq f(1^n)$, and $f(x) = f(1^n)$ for all $x \succ u$. An input $d \in \{0,1\}^n - \{0^n\}$ is called a *min term* if $f(d) \neq f(0^n)$, and $f(x) = f(0^n)$ for all $x \prec d$.

2.2 Communication complexity

Suppose that for a bivariate function $F(x,y)$, the input x is given to Alice and y to Bob. The (deterministic) *communication complexity* $\mathsf{CC}(F)$ is the minimum number of bits needed to be exchanged by the best (deterministic) protocol that computes F (on the worst-case input).

The rank (over $\mathbb{R}$) of the communication matrix for bit-wise composed functions coincides with some natural parameters of the outer function f. For XOR functions $f \circ \oplus$, it holds that $\mathsf{rank}(M_{f\circ\oplus}) = \|\hat{f}\|_0$, and for AND functions $f \circ \wedge$, it holds that $\mathsf{rank}(M_{f\circ\wedge}) = \mathtt{mono}(f)$. When f is OR function of n variables, we have $\mathsf{rank}(M_{f\circ\wedge}) = \mathtt{mono}(\mathsf{OR}_n) = 2^n - 1$.

It is well known that communication can simulate queries. More specifically, for XOR functions and AND functions, we have that

$$\mathsf{CC}(f \circ \wedge) \le 2\mathsf{DT}(f) \text{ and } \mathsf{CC}(f \circ \oplus) \le 2\mathsf{DT}(f). \tag{1}$$

In a {0,1}-communication matrix M, for $b \in \{0,1\}$, a *b-rectangle* is a submatrix of all entries equal to b. The *b-covering number* $\mathsf{Cover}_b(M)$ of matrix M is the minimum number of b-rectangles that can cover all b entries in M. (These b-rectangles need not be disjoint.) For notational convenience, we sometimes write $\mathsf{Cover}_1(F)$ for $\mathsf{Cover}_1(M_F)$. Lovász [36] showed the following bounds.

▶ **Theorem 11** ([36]). *For any Boolean funcion $F(x, y)$, it holds that*

$$\log \mathsf{Cover}_b(M_F) \le \mathsf{CC}(F) \le \log \mathsf{Cover}_b(M_F) \cdot \log \mathsf{rank}(M_F).$$

3 The Sensitivity Conjecture

This section is devoted to the proof of Theorem 1. We will first show the following lemma, in which the first statement is used in this section and the second statement will be used in Section 4 for proving the Log-rank Conjecture of XOR functions.

▶ **Lemma 12.** *For any $f : \{0,1\}^n \to \{0,1\}$, it holds that*
1. $\max\{\mathsf{C}(f, 0^n), \mathsf{C}(f, 1^n)\} \le \mathsf{alt}(f) \cdot \mathsf{s}(f)$
2. $\max\{\mathsf{C}(f, 0^n), \mathsf{C}(f, 1^n)\} \le \mathsf{alt}(f) \cdot \deg_2(f)$.

Proof. First note that it suffices to prove the two upper bounds for $\mathsf{C}(f, 0^n)$, because then we can take $g(x) = f(\bar{x})$ to get that $\mathsf{C}(f, 1^n) = \mathsf{C}(g, 0^n) \le \mathsf{alt}(g) \cdot \mathsf{s}(g) = \mathsf{alt}(f) \cdot \mathsf{s}(f)$.

We prove upper bounds on $\mathsf{C}(f, 0^n)$ by induction on $\mathsf{alt}(f)$. When $\mathsf{alt}(f) = 1$, the function is either monotone or anti-monotone, thus

$$\mathsf{C}(f, 0^n) \le \mathsf{C}(f) = \mathsf{s}(f) \le \deg_2(f),$$

where the first inequality is by definition of $\mathsf{C}(f, 0^n)$, the middle equality is by Fact 7 and the last inequality is because $\mathsf{s}(f) \le \deg_2(f)$ for monotone f (Fact 8). Now we assume that the inequalities in the lemma hold for $\mathsf{alt}(f) < a$ and we will show that they hold for f with $\mathsf{alt}(f) = a$ as well. Let u be a max term of f. Define $S_0(u) \stackrel{\text{def}}{=} \{i \in [n] : u_i = 0\}$, and consider the subcube above u: $\{x : x \succeq u\}$. Let f' be the subfunction obtained by restricting f on this subcube. By the definition of max term $f(u) \ne f(u^i)$ for all $i \in S_0(u)$. Therefore,

$$|S_0(u)| \le \mathsf{s}(f, u) \le \mathsf{s}(f). \tag{2}$$

We know that any point $z \succ u$ has $f(z) = f(1^n) \ne f(u)$. So the number of 1-inputs of f' is odd, implying that $\deg_2(f') = |S_0(u)|$ (Fact 4). Thus we have

$$|S_0(u)| = \deg_2(f') \le \deg_2(f). \tag{3}$$

Now consider another restriction of f, this time to the subcube *under* u, *i.e.* $\{x : x \preceq u\}$. This is implemented by restricting all variables in $S_0(u)$ to 0, yielding a subfunction f'' with

$\mathsf{alt}(f'') \le \mathsf{alt}(f) - 1$. Using induction hypothesis, we have that

$$\begin{aligned}\mathsf{C}(f'', 0^{[n]-S_0(u)}) &\le \mathsf{alt}(f'') \cdot \min\{\mathsf{s}(f''), \deg_2(f'')\} \\ &\le (\mathsf{alt}(f) - 1) \cdot \min\{\mathsf{s}(f), \deg_2(f)\} \end{aligned} \tag{4}$$

Recall that f'' is obtained from f by restricting $|S_0(u)|$ variables, thus

$$\mathsf{C}(f, 0^n) \le |S_0(u)| + \mathsf{C}(f'', 0^{[n]-S_0(u)}).$$

Plugging Eq.(2) and Eq.(4) into the above inequality gives

$$\mathsf{C}(f, 0^n) \le \mathsf{alt}(f) \cdot \min\{\mathsf{s}(f), \deg_2(f)\},$$

completing the induction. ◀

Now we are ready to prove the following theorem, which gives an explicit constant for Theorem 1.

▶ **Theorem 13.** *For any boolean function f,*

$$\mathsf{bs}(f) \le \begin{cases} C_t \cdot \mathsf{s}(f) & \text{if } \mathsf{alt}(f) = 2t, \\ (C_t + 1) \cdot \mathsf{s}(f) & \text{if } \mathsf{alt}(f) = 2t + 1, \end{cases} \tag{5}$$

where $C_t = \sum_{i=1}^{t}(i+2) = \frac{1}{2}t(t+5)$.

Proof. We prove Eq.(5) by induction on $t = \lfloor \mathsf{alt}(f)/2 \rfloor$. Clearly it holds when $t = 0$: If $\mathsf{alt}(f) = 0$ then f is a constant function and $\mathsf{bs}(f) = \mathsf{s}(f) = 0$. When $\mathsf{alt}(f) = 1$, f is monotone or anti-monotone, thus $\mathsf{bs}(f) = \mathsf{s}(f)$.

Now for any Boolean function f with $\mathsf{alt}(f) > 1$, we first consider the case when $\mathsf{alt}(f) = 2t \ge 2$. We will bound the block sensitivity for each input x. Consider the following possible properties for x:

1. there exists a max term u of f such that $x \preceq u$;
2. there exists a min term d of f such that $x \succeq d$.

Case 1: x satisfies at least one of the above conditions. Without loss of generality assume it satisfies the first one; the other case can be similarly argued. Fix such a max term $u \succeq x$. By definition of max term, we know that $\mathsf{alt}(f, u) \le \mathsf{alt}(f) - 1$, and that u is sensitive to all $i \in S_0(u) \stackrel{\text{def}}{=} \{i : u_i = 0\}$. Therefore, $|S_0(u)| \le \mathsf{s}(f, u) \le \mathsf{s}(f)$.

Let f' be the subfunction of f restricted on the subcube $\{t : t \preceq u\}$, then $\mathsf{alt}(f') = \mathsf{alt}(f, u) \le \mathsf{alt}(f) - 1 = 2t - 1 = 2(t-1) + 1$.

By induction hypothesis and the fact that sensitivity is downward non-increasing, we have

$$\mathsf{bs}(f', x) \le \mathsf{bs}(f') \le (C_{t-1} + 1) \cdot \mathsf{s}(f') \le (C_{t-1} + 1) \cdot \mathsf{s}(f). \tag{6}$$

Next it is not hard to see that

$$\mathsf{bs}(f, x) \le \mathsf{bs}(f', x) + |S_0(u)|. \tag{7}$$

Indeed, take any disjoint minimal sensitive blocks $B_1, \ldots, B_\ell \subseteq [n]$ of x (with respect to f), where $\ell = \mathsf{bs}(f, x)$. If $B_i \subseteq [n] - S_0(u)$, then x is still sensitive to B_i in f'. As the B_i's are disjoint, at most $|S_0(u)|$ many B_i's are not contained in $[n] - S_0(u)$, thus at least

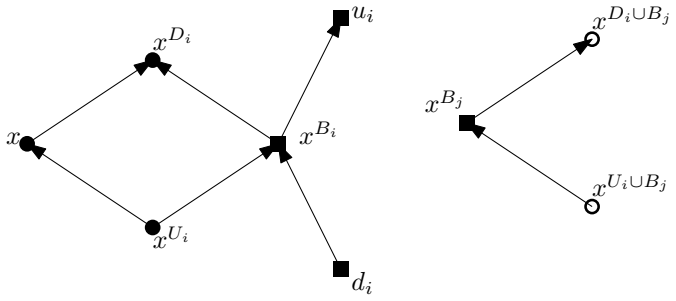


Figure 1 Order among different inputs used in the proof. Arrows indicate the partial order in $\{0,1\}^n$. Solid round circles stand for one Boolean value, and squares stand for the other. The value for hollow circles are not fully determined, but we will show that most of them share the same value with the squares.

$\mathsf{bs}(f,x) - |S_0(u)|$ blocks B_i are still sensitive blocks of x in f'. Therefore, $\mathsf{bs}(f,x) - |S_0(u)| \leq \mathsf{bs}(f',x)$, as Eq.(7) claimed.

Combining Eq.(6), Eq.(7), and the fact that $|S_0(u)| \leq \mathsf{s}(f)$, we conclude that

$$\mathsf{bs}(f,x) \leq \mathsf{bs}(f',x) + |S_0(u)| \leq (C_{t-1}+1)\cdot \mathsf{s}(f') + \mathsf{s}(f) \leq (C_{t-1}+2)\cdot \mathsf{s}(f), \tag{8}$$

which is at most $C_t \cdot \mathsf{s}(f)$ by our setting of parameter $C_t = \sum_{i=1}^{t}(i+2) = C_{t-1} + t + 2$.

Case 2: x satisfies neither of the conditions 1 and 2. So $f(x)$ needs to be the same with both $f(0^n)$ and $f(1^n)$, and f is constant on both subcubes $\{t : t \succeq x\}$ and $\{t : t \preceq x\}$. Otherwise we can take a minimal d where $d \preceq x$ and $f(d) = f(x) \neq f(0^n)$ and by definition d is a min term, or take the maximal u where $u \succeq x$ and $f(u) = f(x) \neq f(1^n)$ and by definition u is a max term.

Fix $\ell = \mathsf{bs}(f,x)$ disjoint minimal sensitive blocks $\{B_1, B_2, \ldots, B_\ell\}$ of x. For each block B_i, decompose it into $B_i = U_i \cup D_i$ where $U_i = \{i \in B_i : x_i = 1\}$ and $D_i = \{i \in B_i : x_i = 0\}$, as depicted below.

$$x = (\overbrace{\underbrace{0\ldots0}_{D_1}\underbrace{1\ldots1}_{U_1}}^{B_1})(\overbrace{\underbrace{0\ldots0}_{D_2}\underbrace{1\ldots1}_{U_2}}^{B_2})\cdots(\overbrace{\underbrace{0\ldots0}_{D_l}\underbrace{1\ldots1}_{U_l}}^{B_l})0\ldots01\ldots1$$

First we will show that for each i, x^{U_i} satisfies condition 1 and x^{D_i} satisfies condition 2, *i.e.* there exist some max term $u \succeq x^{U_i}$ and some min term $d \preceq x^{D_i}$. (See Figure 1 for an illustration.) Indeed, for any sensitive block B_i of x, $f(x^{B_i}) \neq f(x) = f(0^n) = f(1^n)$. Take a *maximal* u_i such that $u_i \succeq x^{B_i}$ and $f(u_i) = f(x^{B_i})$. By definition u_i is a max term. Similarly we can take a min term d_i where $d_i \preceq x^{B_i}$. Then from the definition of U_i and D_i we can conclude that $x^{U_i} \preceq x^{B_i} \preceq u_i$ and $x^{D_i} \succeq x^{B_i} \succeq d_i$. Moreover, both U_i and D_i cannot be empty, since otherwise either $x \preceq x^{D_i} = x^{B_i} \preceq u_i$ or $x \succeq x^{U_i} = x^{B_i} \succeq d_i$, contradicting our assumption of **case 2**. This further indicates that $f(x) = f(x^{U_i}) = f(x^{D_i})$ as we have taken each B_i to be a *minimal* sensitive block.

Next we are going to find some U_i or D_i such that x^{U_i} or x^{D_i} is sensitive to most B_i's. In this case if there are many sensitive blocks of input x, x^{U_i} or x^{D_i} must have high block sensitivity. But we have eliminate this possibility in case 1. To achieve this, we count the following two quantities:

- $\#U$: the number of pairs (i,j) such that $i \neq j$ and $f(x^{U_i}) \neq f(x^{U_i \cup B_j})$,
- $\#D$: the number of pairs (i,j) such that $i \neq j$ and $f(x^{D_i}) \neq f(x^{D_i \cup B_j})$.

Recall that $f(x) = f(x^{U_i}) = f(x^{D_i})$ and $f(x) \neq f(x^{B_j})$, thus it is equivalent to counting

- $\#U$: the number of pairs (i, j) such that $i \neq j$ and $f(x^{B_j}) = f(x^{U_i \cup B_j})$,
- $\#D$: the number of pairs (i, j) such that $i \neq j$ and $f(x^{B_j}) = f(x^{D_i \cup B_j})$.

Now we bound the number of such i's for each j. Fix a block B_j, and consider the subfunction f^u on the subcube $\{z : z \succeq x^{B_j}\}$ and the subfunction f^d on the subcube $\{z : z \preceq x^{B_j}\}$. Let us look at f^u first. Because $D_i \cap B_j = \varnothing$ whenever $i \neq j$, $x^{D_i \cup B_j} \succeq x^{B_j}$ which lies in the domain of f^u. By the definition of certificate complexity of f^u on input x^{B_j}, there is a subcube C of co-dimension $\mathsf{C}(f^u, x^{B_j})$ (with respect to $\{z : z \succeq x^{B_j}\}$) containing x^{B_j}, s.t. f takes a constant 0/1 value on C. Denote by S the set of coordinates in this certificate. Then $S \subseteq \{k \in [n] : (x^{B_j})_k = 0\}$ and $|S| = \mathsf{C}(f^u, x^{B_j})$. Now for each D_i, if $D_i \cap S = \varnothing$, then $f(x^{B_j}) = f(x^{D_i \cup B_j})$ as the values of the certificate variables S are not flipped. As all $\{D_i\}_{i \neq j}$ are disjoint, at most $\mathsf{C}(f^u, x^{B_j})$ many of D_i's may intersect S. Thus $f(x^{B_j}) = f(x^{D_i \cup B_j})$ for all but at most $\mathsf{C}(f^u, x^{B_j})$ many of D_i. Similarly we can say that all but at most $\mathsf{C}(f^d, x^{B_j})$ many of U_i's $(i \neq j)$ satisfy that $f(x^{B_j}) = f(x^{U_i \cup B_j})$. Applying Lemma 12 (statement 1), we have

$$\mathsf{C}(f^u, x^{B_j}) \leq \mathsf{alt}(f^u) \cdot \mathsf{s}(f^u) \leq \mathsf{alt}(f^u) \cdot \mathsf{s}(f),$$

$$\mathsf{C}(f^d, x^{B_j}) \leq \mathsf{alt}(f^d) \cdot \mathsf{s}(f^d) \leq \mathsf{alt}(f^d) \cdot \mathsf{s}(f).$$

Because $\mathsf{alt}(f^u) + \mathsf{alt}(f^d) \leq \mathsf{alt}(f) = 2t$, and there are ℓ sensitive blocks B_i, thus from the second definition of $\#U$ and $\#D$ we can see that

$$\begin{aligned} \#U + \#D &\geq \ell \cdot \big((\ell - 1 - \mathsf{alt}(f^u) \cdot \mathsf{s}(f)) + (\ell - 1 - \mathsf{alt}(f^d) \cdot \mathsf{s}(f))\big) \\ &\geq \ell \cdot 2\,(\ell - 1 - t \cdot \mathsf{s}(f))\,. \end{aligned} \tag{9}$$

Since there are 2ℓ of U_i's and D_i's in total, by pigeonhole principle there must be a T_i (being U_i or D_i) that contributes to at least $(\#U + \#D)/2\ell \geq \ell - 1 - t \cdot \mathsf{s}(f)$ to $(\#U + \#D)$. Fix this T_i. By definition of $\#U$ and $\#D$, there exist at least $\ell - 1 - t \cdot \mathsf{s}(f)$ blocks B_j with $i \neq j$ and $f(x^{T_i}) \neq f(x^{T_i \cup B_j})$. That is, x^{T_i} is sensitive to at least $\ell - 1 - t \cdot \mathsf{s}(f)$ blocks B_j where $j \neq i$. Considering that x^{T_i} is also sensitive to $B_i \backslash T_i$, we conclude that

$$\mathsf{bs}(f, x^{T_i}) \geq 1 + (\ell - 1 - t \cdot \mathsf{s}(f)) = \mathsf{bs}(f, x) - t \cdot \mathsf{s}(f).$$

Finally, recall that we have showed that x^{T_i} satisfies one of the condition 1 and 2. Therefore x^{T_i} is an input falling into **case 1**. By Eq.(8), we have $\mathsf{bs}(f, x^{T_i}) \leq (C_{t-1} + 2) \cdot \mathsf{s}(f)$. Putting everything together, we have

$$\mathsf{bs}(f, x) \leq \mathsf{bs}(f, x^{T_i}) + t \cdot \mathsf{s}(f) \leq (C_{t-1} + 2 + t) \cdot \mathsf{s}(f) = C_t \cdot \mathsf{s}(f).$$

This finishes the proof for $\mathsf{alt}(f) = 2t$.

When $\mathsf{alt}(f) = 2t + 1$, for any input x, $f(x)$ must differ from either $f(0^n)$ or $f(1^n)$ since $f(0^n) \neq f(1^n)$. Without loss of generality, assume that $f(x) \neq f(0^n)$. Take the minimal d such that $d \preceq x$ and $f(d) = f(x) \neq f(0^n)$. By definition d is a min term and x satisfies condition 2. Then using the same analysis above as in **case 1**, we can show $\mathsf{bs}(f, x) \leq (C_t + 1) \cdot \mathsf{s}(f)$ and this finishes the proof. ◀

4 The Log-Rank Conjecture

We prove Theorem 2 and 3 in this section. We start with Theorem 2, which is now easy given Lemma 12. Recall that the second statement of Lemma 12 says that $\max\{\mathsf{C}(f, 0^n), \mathsf{C}(f, 1^n)\} \leq$

$\mathsf{alt}(f) \cdot \deg_2(f)$, therefore

$$\mathsf{C}_{\min}(f) \le \mathsf{alt}(f) \cdot \deg_2(f). \tag{10}$$

As both $\mathsf{alt}(f)$ and $\deg_2(f)$ are downward non-increasing, applying Eq.(10) to all subfunctions of f yields $\mathsf{C}^{\mathrm{clo}}_{\min}(f) \le \mathsf{alt}(f) \cdot \deg_2(f)$. Since $\mathsf{DT}(f) \le \mathsf{C}^{\mathrm{clo}}_{\min}(f) \cdot \deg_2(f)$ (Theorem 6) we get the following.

▶ **Theorem 14.** *For any $f : \{0,1\}^n \to \{0,1\}$, it holds that $\mathsf{DT}(f) \le \mathsf{alt}(f) \cdot \deg_2(f)^2$.*

Theorem 2 follows from this together with the fact that $\mathsf{CC}(f \circ \oplus) \le 2\mathsf{DT}(f)$ (Eq.(1)) and that $\deg_2(f) \le \log \|\hat{f}\|_0 = \log \mathsf{rank}(M_{f\circ\oplus})$ (Fact 5).

Note that if we use the first statement of Lemma 12, we will get the following corollary, which gives better dependence on $\mathsf{alt}(f)$ for low $\mathbb{F}_2$-degree functions.

▶ **Corollary 15.** $\mathsf{DT}(f) \le \mathsf{alt}(f)\mathsf{s}(f) \cdot \deg_2(f)$.

Next we prove Theorem 3 for AND functions. Different than the above approach for XOR functions of going through $\mathsf{DT}(f)$, we directly argue communication complexity of AND functions. Recall that Theorem 3 says that

$$\mathsf{CC}(f \circ \wedge) \le \min\{O(\log^{a+1} \mathsf{rank}(M_{f\circ\wedge})), O(\log^{\frac{a+3}{2}} \mathsf{rank}(M_{f\circ\wedge}) \log n)\}.$$

Proof of Theorem 3. Without loss of generality, we can assume that $f(0^n) = 0$ since otherwise we can compute $\neg f$ first and negate the answer (note that $\mathsf{rank}(M_{\neg f\circ\wedge})$ differs from $\mathsf{rank}(M_{f\circ\wedge})$ by at most 1). For notational convenience let us define $r = \mathtt{mono}(f) = \mathsf{rank}(M_{f\circ\wedge})$ and $\ell = \log r$. For $b \in \{0,1\}$, further define $C_b^{(a)}$ to be the maximum $\mathsf{Cover}_b(f \circ \wedge)$ over all functions $f : \{0,1\}^n \to \{0,1\}$ with alternating number a and $f(0^n) = 0$. We will give three bounds for $C_b^{(a)}$ in terms of $C_b^{(a-1)}$, and combining them gives the claimed result in Theorem 3.

Bound 1, from max terms. We apply this bound for $C_b^{(a)}$ when a and b have different parities, that is, when a is even and $b = 1$, and when a is odd and $b = 0$. Consider the first case and the second is similar. Take any Boolean function f with $f(0^n) = 0$ and $\mathsf{alt}(f) = a$ is even, we have $f(1^n) = 0$. Any 1-input is under some max term, so it is enough to cover inputs under max terms when bounding the $\mathsf{Cover}_1(f)$. Take an arbitrary max term $u \in \{0,1\}^n$. Suppose its Hamming weight is s. Considering the subfunction f' on $\{t : t \succeq u\}$, which is an OR function of $n - s$ variables. In the communication setting, this is the Disjointness function of $n - s$ variables. Thus $\ell = \log \mathsf{rank}(M_{f\circ\wedge}) \ge n - s$. This implies that all max terms u of f are ℓ-close to 1^n in Hamming distance. Considering that different max terms are incomparable by definition, we know that the number of max terms is at most $\binom{n}{\ell}$.

Next we upper bound the 1-rectangles by giving a partition of set of 1-inputs into 1-rectangles. For each max term $u \in \{0,1\}^n$, let $U = \{i \in [n] : u_i = 1\}$, and $k = n - |U|$, then $k \le \ell$. The submatrix $\{(x,y) : x, y \in \{0,1\}^n, x \wedge y \preceq u\}$ is partitioned into 3^k submatrices as follows. Suppose that the set of 0-coordinates in u is $\{i_1, \ldots, i_k\}$, then for each i_j, we can choose (x_{i_j}, y_{i_j}) from the set $\{(0,0),(0,1),(1,0)\}$ to enforce $x_{i_j} \wedge y_{i_j} = 0$. Thus there are 3^k ways of restricting these k variables in $\bar{U}$, giving 3^k submatrices. Let $f_u : \{0,1\}^U \to \{0,1\}$ be the subfunction of f restricted on the subcube $\{t : t \preceq u\}$ where $f_u(z_U) = f(z_U, 0_{\bar{U}})$. (Here the input to f is $x' \wedge y'$ at U and 0 at $\bar{U}$.) Note that each of the 3^k submatrices is still the communication matrix of $f_u \circ \wedge$ for some max term u. Also note that this f_u has $f_u(0_U) = 0$,

but $f_u(1_U) = 1$ and $\mathsf{alt}(f_u) \leq \mathsf{alt}(f) - 1$. Since all the 1-inputs of f are under some max term u, the 1-covering number $\mathsf{Cover}^1(f \circ \wedge)$ can be upper bounded by the following:

$$\mathsf{Cover}_1(f \circ \wedge) \leq \sum_{u:\text{max term}} 3^\ell \cdot \mathsf{Cover}_1(f_u \circ \wedge) \leq \binom{n}{\ell} \cdot 3^\ell \cdot \max_{u:\text{max term}} \mathsf{Cover}_1(f_u \circ \wedge).$$

Using the fact $\mathsf{alt}(f_u) \leq \mathsf{alt}(f) - 1$, and that the above inequality holds for any f, we have the following bound on $C_1^{(a)}$:

$$\log C_1^{(a)} \leq 3\ell \cdot \log n + \log C_1^{(a-1)}, \text{ when } a \text{ is even.} \tag{11}$$

Similarly, when a is odd, $f(1^n) = 1$, and thus any 0-input is under some max term. A similar argument shows the following bound on $C_0^{(a)}$:

$$\log C_0^{(a)} \leq 3\ell \cdot \log n + \log C_0^{(a-1)}, \text{ when } a \text{ is odd.} \tag{12}$$

Bound 2, from min terms. Take any Boolean function f with $f(0^n) = 0$. Then any 1-input must be above some min term. Take any min term d. Let $D = \{i : d_i = 1\}$. If we restrict variables x_i and y_i to 1 for all $i \in D$, then we go to a rectangle $\{(x, y) : x_i = y_i = 1, \forall i \in D\}$. The union of these rectangles for all min terms d contains all 1-inputs. Restrict f on the subcube $\{z : z \succeq d\}$ to get a subfunction f_d, which has $f_d(0^{\bar{D}}) = 1$, and $\mathsf{alt}(f_d) \leq \mathsf{alt}(f) - 1$. Note that for each min term d, we have $\alpha(d) = \sum_{x \preceq d} (-1)^{|d \oplus x|} f(x) = 1 \neq 0$ [3], which contributes 1 to $\mathtt{mono}(f)$, thus the number of min terms is at most $\mathtt{mono}(f) = r$. Since each 1-input of f is above some min term d, the 1-covering number $\mathsf{Cover}_1(f)$ has

$$\mathsf{Cover}_1(f \circ \wedge) \leq \sum_{d:\text{min term}} \mathsf{Cover}_1(f_d \circ \wedge) \leq r \cdot \max_{d:\text{min term}} \mathsf{Cover}_1(f_d \circ \wedge).$$

Note that $\mathsf{alt}(f_d) \leq \mathsf{alt}(f) - 1$, and f_d takes value 1 on its all-0 input, thus $\mathsf{Cover}_1(f_d \circ \wedge) = \mathsf{Cover}_0(\neg f_d \circ \wedge) \leq C_0^{(a-1)}$ (note that the maximum in the definition of C_0 is over all f with $f(0^n) = 0$). This implies

$$\log C_1^{(a)} \leq \ell + \log C_0^{(a-1)}. \tag{13}$$

Note that this inequality holds as long as $f(0^n) = 0$, regardless of the parity of a.

Bound 3, from CC. When a is odd, we have a bound for $C_0^{(a)}$ by Eq.(12) and a bound for $C_1^{(a)}$ by Eq.(13). When a is even, we have two bounds for $C_1^{(a)}$, Eq.(11) and Eq.(13), but no bound for $C_0^{(a)}$. Note that we can always use CC to bound $C_0^{(a)}$:

$$\begin{aligned}\log \mathsf{Cover}_0(f \circ \wedge) &\leq \mathsf{CC}(f \circ \wedge) \\ &\leq \log \mathsf{rank}(M_{f \circ \wedge}) \cdot \log \mathsf{Cover}_1(f \circ \wedge) \\ &= \ell \cdot \log \mathsf{Cover}_1(f \circ \wedge),\end{aligned}$$

This implies that

$$\log C_0^{(a)} \leq \ell \cdot \log C_1^{(a)}. \tag{14}$$

[3] If $f(0^n) = 1$, then for each min term d, we have $\alpha(d) = \sum_{x \preceq d} (-1)^{|d \oplus x|} f(x) = -1$, which is still non-zero.

Similarly it also holds that $\log C_1^{(a)} \le \ell \cdot \log \mathsf{C}_0^{(a)}$.

Now we combine the three bounds and prove the theorem by induction on a. In the base case of $a = 0$, the function is constant 0 and thus $C_0^{(0)} = 1$ and $C_1^{(0)} = 0$. For general a, we can repeatedly apply Eq.(13) and Eq.(14) to get

$$\log C_1^{(a)} \le \sum_{i=1}^{a} \ell^i = (1 + o(1))\ell^a.$$

Thus $\mathsf{CC}(f \circ \wedge) \le \ell \cdot \log C_1^{(a)} \le (1 + o(1))\ell^{a+1}$.

If we can tolerate a $\log n$ factor, then the dependence on a can be made slightly better. Assume that a is even, we have

$$\begin{aligned}
\log C_1^{(a)} &\le \ell + \log C_0^{(a-1)} && \text{(by Eq.(13))} \\
&\le \ell + 3\ell \log n + \log C_0^{(a-2)} && \text{(by Eq.(12))} \\
&\le \ell + 3\ell \log n + \ell \log C_1^{(a-2)}. && \text{(by Eq.(14))}
\end{aligned}$$

Solving this recursion gives $\log C_1^{(a)} \le O(\ell^{\frac{a}{2}} \log n)$, and thus $\mathsf{CC} = O(\ell^{\frac{a}{2}+1} \log n)$. When a is odd, we can use Eq.(13) and Eq.(14) to reduce it to the "even a" case, resulting a bound $\mathsf{CC} \le O(\ell^{\frac{a+3}{2}} \log n)$. Putting these two cases together, we get the claimed bound. ◀

Acknowledgement. The authors would like to thank Xin Huang for valuable discussions.

References

1. Andris Ambainis. Communication complexity in a 3-computer model. *Algorithmica*, 16(3):298–301, 1996.
2. Andris Ambainis, Mohammad Bavarian, Yihan Gao, Jieming Mao, Xiaoming Sun, and Song Zuo. Tighter relations between sensitivity and other complexity measures. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, pages 101–113. 2014.
3. Andris Ambainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Sensitivity versus certificate complexity of Boolean functions. *arXiv preprint arXiv:1503.07691*, 2015.
4. Andris Ambainis, Leonard Schulman, Amnon Ta-Shma, Umesh Vazirani, and Avi Wigderson. The quantum communication complexity of sampling. *SIAM Journal on Computing*, 32(6):1570–1585, 2003.
5. Andris Ambainis and Xiaoming Sun. New separation between $s(f)$ and $bs(f)$. *CoRR, abs/1108.3494*, 2011.
6. Andris Ambainis and Jevgēnijs Vihrovs. Size of sets with small sensitivity: A generalization of Simon's lemma. In *Theory and Applications of Models of Computation*, pages 122–133. Springer, 2015.
7. László Babai and Peter G. Kimmel. Randomized simultaneous messages: Solution of a problem of Yao in communication complexity. In *IEEE Conference on Computational Complexity*, pages 239–246, 1997.
8. Mitali Bafna, Satyanarayana V Lokam, Sébastien Tavenas, and Ameya Velingker. On the sensitivity conjecture for read-k formulas. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 58. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
9. Shalev Ben-David. Low-sensitivity functions from unambiguous certificates. *arXiv preprint arXiv:1605.07084*, 2016.

10 Eric Blais, Clément L Canonne, Igor C Oliveira, Rocco A Servedio, and Li-Yang Tan. Learning circuits with few negations. *arXiv:1410.8420*, 2014.

11 Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16), 2001.

12 Harry Buhrman and Ronald de Wolf. Communication complexity lower bounds by polynomials. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, pages 120–130, 2001.

13 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

14 Sourav Chakraborty. On the sensitivity of cyclically-invariant Boolean functions. In *Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on*, pages 163–167. IEEE, 2005.

15 Arkadev Chattopadhyay and Toniann Pitassi. The story of set disjointness. *SIGACT News*, 41(3):59–85, 2010.

16 Dmitry Gavinsky, Julia Kempe, and Ronald de Wolf. Quantum communication cannot simulate a public coin. *arXiv:quant-ph/0411051*, 2004.

17 Dmitry Gavinsky and Shachar Lovett. En route to the log-rank conjecture: New reductions and equialent formulations. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, pages 514–524. 2014.

18 Justin Gilmer, Michal Kouckỳ, and Michael E Saks. A new approach to the sensitivity conjecture. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 247–254. ACM, 2015.

19 Mika Göös and T. S. Jayram. A composition theorem for conical juntas. *Electronic Colloquium on Computational Complexity*, 22:167, 2015.

20 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science*, pages 1077–1088, 2015.

21 Parikshit Gopalan, Noam Nisan, Rocco A Servedio, Kunal Talwar, and Avi Wigderson. Smooth Boolean functions are easy: efficient algorithms for low-sensitivity functions. *arXiv preprint arXiv:1508.02420*, 2015.

22 Parikshit Gopalan, Rocco Servedio, Avishay Tal, and Avi Wigderson. Degree and sensitivity: tails of two distributions. *arXiv preprint arXiv:1604.07432*, 2016.

23 Siyao Guo, Tal Malkin, Igor C Oliveira, and Alon Rosen. The power of negations in cryptography. In *Theory of Cryptography*, pages 36–65. Springer, 2015.

24 Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. Variations on the sensitivity conjecture. (4):1–27, 2011.

25 Kun He, Qian Li, and Xiaoming Sun. A tighter relation between sensitivity and certificate complexity. *arXiv preprint arXiv:1609.04342*, 2016.

26 Wei Huang, Yaoyun Shi, Shengyu Zhang, and Yufan Zhu. The communication complexity of the Hamming Distance problem. *Information Processing Letters*, 99(4):149–153, 2006.

27 Rahul Jain, Hartmut Klauck, and Shengyu Zhang. Depth-independent lower bounds on the communication complexity of read-once Boolean formulas. In *Proceedings of the 16th Annual International Conference on Computing and Combinatorics*, pages 54–59, 2010.

28 T. S. Jayram, Swastik Kopparty, and Prasad Raghavendra. On the communication complexity of read-once AC^0 formulae. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity*, pages 329–340, 2009.

29 Stasys Jukna. *Boolean Function Complexity*. Springer, 2012.

30 Raghav Kulkarni and Miklos Santha. Query complexity of matroids. In *Proceedings of the 8th International Conference on Algorithms and Complexity*, 2013.

31 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, Cambridge, UK, 1997.

32 Troy Lee and Shengyu Zhang. Composition theorems in communication complexity. In *Automata, Languages and Programming, 37th International Colloquium*, pages 475–489, 2010.

33 Nikos Leonardos and Michael E. Saks. Lower bounds on the randomized communication complexity of read-once functions. *Computational Complexity*, 19(2):153–181, 2010.

34 Ming Lam Leung, Yang Li, and Shengyu Zhang. Tight bounds on the communication complexity of symmetric XOR functions in one-way and SMP models. In *Proceedings of the 8th Annual Conference on Theory and Applications of Models of Computation*, pages 403–408, 2011.

35 Yang Liu and Shengyu Zhang. Quantum and randomized communication complexity of XOR functions in the SMP model. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:10, 2013.

36 László Lovász. Communication complexity — a survey. In Bernhard Korte, Laszlo Lovasz, Hans Jurgen Promel, and Alexander Schrijver, editors, *Paths, Flows, and VLSI Layout*. Oxford University Press, 1990.

37 László Lovász and Michael E. Saks. Lattices, Möbius functions and communication complexity. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 81–90, 1988.

38 László Lovász and Michael E. Saks. Communication complexity and combinatorial lattice theory. *Journal of Computer and System Sciences*, 47(2):322–349, 1993.

39 Shachar Lovett. Communication is bounded by root of rank. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 842–846, 2014.

40 AA Markov. On the inversion complexity of a system of functions. *Journal of the ACM (JACM)*, 5(4):331–334, 1958.

41 Kurt Mehlhorn and Erik M. Schmidt. Las Vegas is better than determinism in VLSI and distributed computing (extended abstract). In *Proceedings of the 14th annual ACM symposium on Theory of computing*, pages 330–337, 1982.

42 Ashley Montanaro and Tobias Osborne. On the communication complexity of XOR functions, 2010. http://arxiv.org/abs/0909.3392v2.

43 Hiroki Morizumi. Limiting negations in formulas. In *Automata, Languages and Programming*, pages 701–712. Springer, 2009.

44 Hiroki Morizumi. Limiting negations in non-deterministic circuits. *Theoretical Computer Science*, 410(38):3988–3994, 2009.

45 Hiroki Morizumi. Sensitivity, block sensitivity, and certificate complexity of unate functions and read-once functions. In *Theoretical Computer Science*, pages 104–110. Springer, 2014.

46 Ilan Newman and Mario Szegedy. Public vs. private coin flips in one round communication games. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 561–570, 1996.

47 Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991.

48 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

49 Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995.

50 Ryan O'Donnell, John Wright, Yu Zhao, Xiaorui Sun, and Li-Yang Tan. A composition theorem for parity kill number. In *Proceedings of the 29th Conference on Computational Complexity*, pages 144–154, 2014.

51 David Rubinstein. Sensitivity vs. block sensitivity of Boolean functions. *Combinatorica*, 15(2):297–299, 1995.

52 Miklos Santha and Christopher Wilson. Limiting negations in constant depth circuits. *SIAM Journal on Computing*, 22(2):294–302, 1993.

53 Alexander A. Sherstov. Communication complexity theory: Thirty-five years of set disjointness. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium*, pages 24–43, 2014.

54 Hans-Ulrich Simon. A tight $\Omega(\log\log n)$-bound on the time for parallel ram's to compute nondegenerated Boolean functions. In *Proceedings of the 1983 International Conference on Fundamentals of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 439–444, 1983.

55 Xiaoming Sun and Chengu Wang. Randomized communication complexity for linear algebra problems over finite fields. In *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science*, pages 477–488, 2012.

56 Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: an extension of Markov's theorem. 2003.

57 Sébastien Tavenas and C. S. Karthik. On the sensitivity conjecture for disjunctive normal forms. In *Proceedings of the 36th Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 15:1–15:15, 2016.

58 Hing Yin Tsang. On Boolean functions with low sensitivity. *manuscript*, 2015. available at http://theorycenter.cs.uchicago.edu/REU/2014/final-papers/tsang.pdf.

59 Hing Yin Tsang, Chung Hoi Wong, Ning Xie, and Shengyu Zhang. Fourier sparsity, spectral norm, and the Log-rank Conjecture. In *Proceedings of the 54th Annual IEEE Symposium Foundations of Computer Science*, pages 658–667, 2013.

60 György Turán. The critical complexity of graph properties. *Information Processing Letters*, 18(3):151–153, 1984.

61 Paul Valiant. The log-rank conjecture and low degree polynomials. *Information Processing Letters*, 89(2):99–103, 2004.

62 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

63 Andrew Chi-Chih Yao. On the power of quantum fingerprinting. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 77–81, 2003.

64 Penghui Yao. Parity decision tree complexity and 4-party communication complexity of xor-functions are polynomially equivalent. *arXiv:*, 1506.02936, 2015.

65 Shengyu Zhang. Quantum strategic game theory. In *Proceedings of the 3rd Innovations in Theoretical Computer Science*, pages 39–59, 2012.

66 Shengyu Zhang. Efficient quantum protocols for XOR functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1878–1885, 2014.

67 Zhiqiang Zhang and Yaoyun Shi. Communication complexities of symmetric XOR functions. *Quantum Information & Computation*, 9(3):255–263, 2009.

Randomized Communication vs. Partition Number*

Mika Göös[1], T. S. Jayram[2], Toniann Pitassi[3], and Thomas Watson[4]

1 **Harvard University, Cambridge, MA, USA**
mika@seas.harvard.edu
2 **IBM Almaden, San Jose, CA, USA**
jayram@us.ibm.com
3 **University of Toronto, Toronto, Canada**
toni@cs.toronto.edu
4 **University of Memphis, Memphis, TN, USA**
thomas.watson@memphis.edu

Abstract

We show that *randomized* communication complexity can be superlogarithmic in the partition number of the associated communication matrix, and we obtain near-optimal *randomized* lower bounds for the Clique vs. Independent Set problem. These results strengthen the deterministic lower bounds obtained in prior work (Göös, Pitassi, and Watson, FOCS 2015). One of our main technical contributions states that information complexity when the cost is measured with respect to only 1-inputs (or only 0-inputs) is essentially equivalent to information complexity with respect to all inputs.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases communication complexity, partition number, information complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.52

1 Introduction

A prior work [16] exhibited a boolean function $F\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ whose deterministic communication complexity is superlogarithmic in the *partition number*

$$\chi(F) \;\coloneqq\; \chi_0(F) + \chi_1(F)$$

where $\chi_i(F)$ is the least number of rectangles (sets of the form $A \times B$ where $A \subseteq \mathcal{X}$, $B \subseteq \mathcal{Y}$) needed to partition the set $F^{-1}(i)$. In this follow-up work, we upgrade the lower-bound results from [16] to hold against randomized protocols – here the notation $\tilde{\Omega}(m)$ hides factors polylogarithmic in m.

▶ **Theorem 1.** *There is an F with randomized communication complexity $\tilde{\Omega}(\log^{1.5} \chi(F))$.*

▶ **Theorem 2.** *There is an F with randomized communication complexity $\tilde{\Omega}(\log^{2} \chi_1(F))$.*

A main technical contribution of our paper – which is key to both the proofs of Theorem 1 as well as the subsequent strengthening by [5] – informally states that the information complexity of a function (as defined by [9]) remains essentially unchanged if the cost is

* The full version of this work is available at [14], https://eccc.weizmann.ac.il/report/2015/169/.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 52; pp. 52:1–52:15

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

measured with respect to only 1-inputs (or only 0-inputs) rather than all inputs. We say a protocol Π is ϵ-correct if it succeeds with probability at least $1-\epsilon$ on each input, and define $\mathrm{IC}(\Pi)$ as the maximum over all input distributions of the information cost (defined later), and define $\mathrm{IC}^b(\Pi)$ similarly but with the maximum over all distributions over b-inputs (for some $b \in \{0,1\}$).

▶ **Theorem 3.** *Fix any F and $b \in \{0,1\}$. For every 1/3-correct protocol Π there is a 1/3-correct protocol Π' such that $\mathrm{IC}(\Pi') \leq O\big(\mathrm{IC}^b(\Pi) + \log(\mathrm{CC}(\Pi)+2)\big)$. Moreover, $\mathrm{CC}(\Pi') \leq O\big(\mathrm{CC}(\Pi)\cdot\log(\mathrm{CC}(\Pi)+2)\big)$.*

In the theorem statement above, the additional lower order term involving the communication cost appears due to technical reasons. This makes the statement slightly weaker but this is mitigated in the aforementioned applications due to the additional fact that we can also bound the communication cost of the new protocol.

1.1 Applications and discussion

Theorem 1

Prior to this work, no examples of F were known with randomized communication complexity larger than $\log\chi(F)$. In fact, such a separation cannot be obtained using the usual rectangle-based lower-bound methods, as catalogued by Jain and Klauck [17]. In particular, Theorem 1 shows that randomized complexity can be polynomially larger than the *partition bound* [17, 19], which is one of the most powerful general lower bound methods for randomized communication. (Consequently, our proof of Theorem 1 has to exploit another powerful lower-bound method, namely *information complexity*.) Note also that every F has deterministic communication complexity at least $\log\chi(F)$ and at most $O(\log^2\chi(F))$, where the latter upper bound is a classical result of [2]. Theorem 1 shows that the upper bound cannot be improved much even if we allow randomization.

Theorem 2

The relationship between $\chi_1(F)$ and the communication complexity of F can be equivalently formulated in the language of the *Clique vs. Independent Set* game, played on a graph derived from F (Alice holds a clique, Bob holds an independent set: do they intersect?). See [34, §4] or [21, §4.4] for the equivalence. Yannakakis [34] (extending [2]) proved that every F has deterministic communication complexity at most $O(\log^2\chi_1(F))$. Our Theorem 2 shows that this upper bound is essentially tight even if we allow randomized protocols, and it implies that there is a graph on n nodes for which Clique vs. Independent Set requires $\tilde{\Omega}(\log^2 n)$ randomized communication. (The deterministic upper bound $O(\log^2 n)$ holds for all graphs.)

Extension complexity. In fact, we prove Theorem 2 by showing that (the negation of) the function F has high *approximate nonnegative rank* (a.k.a. smooth rectangle bound; see Section 2 for definitions). One consequence in the field of *extended formulations* (see [34, 11] for definitions) is that we obtain a graph G such that the polytope generated by the so-called "clique inequalities" of G has extension complexity $n^{\tilde{\Omega}(\log n)}$. (The slack matrix associated with the clique inequalities is simply (the negation of) the Clique vs. Independent Set game. These inequalities capture the independent set polytope of G when G is perfect – our graph G however is not.) The previous bound in this direction was $n^{\Omega(\log^{0.128} n)}$ from a related work [13]. Technically speaking, the lower bound from [13] was proved for *nondeterministic* communication complexity, so the full result remains incomparable with Theorem 2.

Log-rank conjecture. The famous log-rank conjecture of Lovász and Saks [30] postulates that the deterministic communication complexity of F is polynomially related to $\log \operatorname{rank}(F)$. Gavinsky and Lovett [12] have shown that the conjecture is equivalent to asking whether the randomized communication complexity of F is polynomially bounded in $\log \operatorname{rank}(F)$. Here our Theorem 2 gives at least a near-quadratic separation between the randomized communication complexity of F and $\log \operatorname{rank}(F) \leq \log \chi_1(F)$; the previous best lower bound was $\Omega(\log^{1.63} \operatorname{rank}(F))$ due to Kushilevitz [26]. Furthermore, Troy Lee has pointed out to us that our construction underlying Theorem 2 exhibits nearly a 4-th power separation between the logarithms of *approximate nonnegative rank* and *approximate rank*. This gives lower bounds for the so-called *log-approximate-rank conjecture* [28, Conjecture 42], which is the randomized analogue of the log-rank conjecture. The previous best separation was quadratic (as witnessed by the set-disjointness problem).

Theorem 3

One-sided information complexity satisfies a famous direct sum property ([6, 9]): for any protocol Π computing $\mathrm{AND}_k \circ F^k$ (i.e., the AND of k copies of F) there exists a protocol Π' computing F with $\mathrm{IC}^1(\Pi') \leq O(\mathrm{IC}^1(\Pi)/k)$ (see, e.g., [5, Claim 37]). One can also formulate a dual lemma for $\mathrm{OR}_k \circ F^k$ in terms of IC^0. This is the context where our Theorem 3 relating IC and IC^1 (and IC^0) is useful: it implies that analogous direct sum lemmas hold for *two-sided* information complexity, up to low order terms. Iterating such a two-sided lemma some constantly many times, one obtains an alternative proof for the result that every n-bit constant-depth balanced read-once AND–OR tree with binary bottom fan-in (defining an Alice–Bob bipartition of input bits) has randomized communication complexity $\Omega(n)$; this result was first proved in [20, 29] even for *unbalanced* trees.

Another application of Theorem 3 appears in the recent work [5]. They improved our 1.5-th power separation in Theorem 1 to near-quadratic (which is optimal) by iteratively applying Theorem 3 to analyze a communication analogue of a query-complexity construction due to Ambainis, Kokainis, and Kothari [4] (which is a variation of usual AND–OR trees).

1.2 Our techniques

The basic strategy in [16] for obtaining the deterministic versions of Theorems 1–2 was to first obtain analogous gaps in the easier-to-understand world of query complexity, then "lift" the results to communication complexity using a so-called *simulation lemma*. For getting randomized lower bounds, two obstacles immediately present themselves: (i) The functions studied in [16] are too easy for randomized protocols (as shown by [31]). (ii) There is no known simulation lemma for the bounded-error randomized setting.

To handle obstacle (i), we modify the functions from [16] in a way that preserves their low partition numbers while eliminating the structure that was exploitable by randomized protocols. (Similar constructions have been given by [3, 1].) To handle obstacle (ii) for Theorem 2, we actually prove a lower bound for a model that is stronger than the standard randomized model, but for which there *is* a known simulation lemma [15]. This idea alone does not handle obstacle (ii) for Theorem 1, though. For that, we start by giving a proof of the query complexity analogue of Theorem 1, then develop a way to *mimic* that argument using communication complexity, by going through information complexity (exploiting machinery from [23] and [10]). In the process, this yields our Theorem 3 (one-sided is equivalent to two-sided information complexity), which is of independent interest.

2 Complexity Measures

We study the following communication complexity models/measures; see Figure 1. For any complexity measure $\mathcal{C}$ we write $\mathsf{co}\mathcal{C}(F) \coloneqq \mathcal{C}(\neg F)$ and $2\mathcal{C}(F) \coloneqq \max\{\mathcal{C}(F), \mathsf{co}\mathcal{C}(F)\}$ for short.

- **P^{cc}:** The deterministic communication complexity of F is denoted $\mathsf{P}^{\mathsf{cc}}(F)$.
- **$\mathsf{BPP}^{\mathsf{cc}}$:** The randomized communication complexity of F is denoted $\mathsf{BPP}^{\mathsf{cc}}(F)$.
- **$\mathsf{UP}^{\mathsf{cc}}$:** Recall (e.g., [27, 21]) that a cost-c nondeterministic protocol for F corresponds to a covering (allowing overlaps) of $F^{-1}(1)$ with 2^c rectangles. A nondeterministic protocol is *unambiguous* if on every 1-input there is a unique accepting computation; combinatorially, this means we have a disjoint covering (partition) of $F^{-1}(1)$. We define $\mathsf{UP}^{\mathsf{cc}}(F) \coloneqq \lceil \log \chi_1(F) \rceil$. Thus $\mathsf{coUP}^{\mathsf{cc}}(F) = \lceil \log \chi_0(F) \rceil$, and $2\mathsf{UP}^{\mathsf{cc}}(F) \in \lceil \log \chi(F) \rceil \pm 1$.
- **$\mathsf{WAPP}^{\mathsf{cc}}$:** Abstractly speaking, a WAPP computation (*Weak Almost-Wide* PP; introduced in [8]) is a randomized computation that accepts 1-inputs with probability in $[(1-\epsilon)\alpha, \alpha]$, and 0-inputs with probability in $[0, \epsilon\alpha]$, where $\epsilon < 1/2$ is an error parameter and $\alpha = \alpha(n) > 0$ is arbitrary.
  Instantiating this for protocols, we define $\mathsf{WAPP}^{\mathsf{cc}}_{\epsilon}(F)$ as the least "cost" of a randomized (public-coin) protocol Π that computes F in the above sense; the "cost" of a protocol Π with parameter α is defined as the usual communication cost (number of bits communicated) plus $\log(1/\alpha)$. In this definition, we may assume w.l.o.g. that Π is *zero-communication* [23]: Π is simply a probability distribution over rectangles R, and Π accepts an input (x, y) iff $(x, y) \in R$ for the randomly chosen R. Such a protocol Π exchanges only 2 bits to check the condition $(x, y) \in R$, and the rest of the cost is coming from having a tiny α.
  We note that $\mathsf{WAPP}^{\mathsf{cc}}$ corresponds to the (one-sided) *smooth rectangle bound* of [17], which is known to be equivalent to *approximate nonnegative rank* [24]. A consequence of this equivalence is that $\mathsf{WAPP}^{\mathsf{cc}}$ could alternatively be defined without charging anything for $\alpha > 0$, as long as we restrict our protocols to be *private-coin*; see also [15, Theorem 9]. Also, $2\mathsf{WAPP}^{\mathsf{cc}}$ is equivalent to the *relaxed partition bound* of [23] (we elaborate on this in Section 4.2). We remark that $\mathsf{WAPP}^{\mathsf{cc}}$ is not amenable to efficient amplification of the error parameter; there can be an exponential gap between $\mathsf{WAPP}^{\mathsf{cc}}_{\epsilon}$ and $\mathsf{WAPP}^{\mathsf{cc}}_{\delta}$ for different constants ϵ and δ, at least for partial functions [15, Theorem 6].

Define the following decision tree models/measures for a boolean function $f\colon \{0,1\}^n \to \{0,1\}$:

- **P^{dt}:** The deterministic decision tree complexity of f is denoted $\mathsf{P}^{\mathsf{dt}}(f)$.
- **$\mathsf{BPP}^{\mathsf{dt}}$:** The randomized decision tree complexity of f is denoted $\mathsf{BPP}^{\mathsf{dt}}(f)$.
- **$\mathsf{UP}^{\mathsf{dt}}$:** A nondeterministic decision tree is a DNF formula. We think of the conjunctions in the DNF formula as *certificates* – partial assignments to inputs that force the function to be 1. The cost is the maximum number of input bits read by a certificate. A nondeterministic decision tree is *unambiguous* if on every 1-input there is a unique accepting certificate. We define $\mathsf{UP}^{\mathsf{dt}}(f)$ as the least cost of an unambiguous decision tree for f. Other works that have studied unambiguous decision trees include [33, 7, 13, 16, 25].
- **$\mathsf{WAPP}^{\mathsf{dt}}$:** We define $\mathsf{WAPP}^{\mathsf{dt}}_{\epsilon}(f)$ as the least height of a randomized decision tree that accepts 1-inputs with probability in $[(1-\epsilon)\alpha, \alpha]$, and 0-inputs with probability in $[0, \epsilon\alpha]$, where $\alpha = \alpha(n) > 0$ is arbitrary. (Note that only the number of queries matters; we do not charge for α being small.) Like the communication version, this measure is not amenable to efficient amplification of the error parameter [15].
  The analogue of a $\mathsf{WAPP}^{\mathsf{cc}}$ protocol being w.l.o.g. a distribution over rectangles is that a $\mathsf{WAPP}^{\mathsf{dt}}$ decision tree is w.l.o.g. a distribution over conjunctions. This implies that we may characterize $\mathsf{WAPP}^{\mathsf{dt}}_{\epsilon}(f)$ using *conical juntas*: A *conical junta* h is a nonnegative

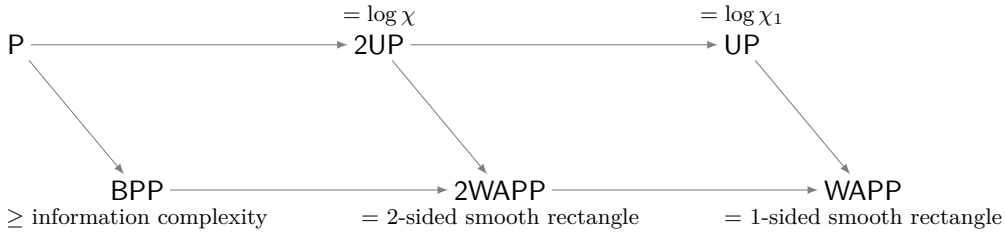


Figure 1 Models of computation that can be instantiated for both communication and query complexity. Here A ⟶ B means that model B can simulate model A without any overhead.

linear combination of conjunctions. That is, $h = \sum w_C C$ where the sum ranges over conjunctions $C\colon \{0,1\}^n \to \{0,1\}$ and $w_C \geq 0$ for all C. Then $\mathsf{WAPP}^{\mathsf{dt}}_{\epsilon}(f)$ is the least *degree* (maximum width of a conjunction with positive weight in h) of a conical junta h that ϵ-approximates f in the sense that $h(z) \in [1-\epsilon, 1]$ for all $z \in f^{-1}(1)$, and $h(z) \in [0, \epsilon]$ for all $z \in f^{-1}(0)$. Other works have studied conical juntas under such names as the (one-sided) *partition bound for query complexity* [17] and *query complexity in expectation* [22].

3 Overview

In this section we give an outline for obtaining our main results, Theorems 1–2. For complexity models/measures $\mathcal{C}$ and $\mathcal{C}'$, we informally say "$\mathcal{C}$-vs-$\mathcal{C}'$ gap" to mean the existence of a function whose $\mathcal{C}$ complexity is significantly higher than its $\mathcal{C}'$ complexity. Using the notation defined in Section 2, we can rephrase our main results as follows.

▶ **Theorem 1 ($\mathsf{BPP}^{\mathsf{cc}}$-vs-$\mathsf{2UP}^{\mathsf{cc}}$).** *There is an F such that* $\mathsf{BPP}^{\mathsf{cc}}(F) \geq \tilde{\Omega}(\mathsf{2UP}^{\mathsf{cc}}(F)^{1.5})$.

▶ **Theorem 2 ($\mathsf{BPP}^{\mathsf{cc}}$-vs-$\mathsf{UP}^{\mathsf{cc}}$).** *There is an F such that* $\mathsf{BPP}^{\mathsf{cc}}(F) \geq \tilde{\Omega}(\mathsf{UP}^{\mathsf{cc}}(F)^{2})$.

1. **Tribes-List (Section 3.1):** Our starting point is to define *Tribes-List*, a variant of a function introduced in [16]. Its purpose is to witness a BPP-vs-UP gap for query complexity.
2. **Composition (Section 3.2):** Next, we modify Tribes-List using two types of function composition, which we call *lifting* and AND-*composition*, to obtain candidate functions for BPP-vs-2UP gaps in both query and communication complexity.
3. **Overview of proofs (Section 3.3):** With the candidate functions defined, we outline our strategy to prove the desired communication lower bounds.

3.1 Tribes-List

The *Tribes-List* function $\mathrm{TL}\colon \{0,1\}^n \to \{0,1\}$ is defined on $n \coloneqq \Theta(k^3 \log k)$ bits where k is a parameter. We think of the input as a $k \times k$ matrix M with entries M_{ij} taking values from the alphabet $\Sigma \coloneqq \{0,1\} \times ([k]^{k-1} \cup \{\perp\})$. Here each entry is encoded with $\Theta(k \log k)$ bits, and we assume that the encoding of $M_{ij} = (m_{ij}, p_{ij}) \in \Sigma$ is such that a single bit is used to encode the value $m_{ij} \in \{0,1\}$ and another bit is used to encode whether or not $p_{ij} = \perp$. If $p_{ij} \neq \perp$, then we can learn its exact value in $[k]^{k-1}$ by querying all the $\Theta(k \log k)$ bits.

Informally, we have $\mathrm{TL}(M) = 1$ iff M has a unique all-$(1,*)$ column (here $*$ is a wildcard) that also contains an entry with $k-1$ pointers to entries of the form $(0,*)$ in all other

Unambiguous decision tree for TL:

Nondeterministically guess a column index $j \in [k]$. Consider the entries $M_{ij} = (m_{ij}, p_{ij})$ for $i \in [k]$: check that $m_{ij} = 1$ for all i and that $p_{ij} \neq \bot$ for at least one i (this is $\leq 2k$ queries). Let i be the first row index for which $p_{ij} \neq \bot$ and read the full value of p_{ij} (this is $\Theta(k \log k)$ queries). Interpret $p_{ij} \in [k]^{[k] \setminus \{j\}}$ as a *list of pointers*, describing a row index for all columns other than j. For each of these $k-1$ pointed-to entries $M_{i'j'}$, check that $m_{i'j'} = 0$ (this is $k-1$ queries).

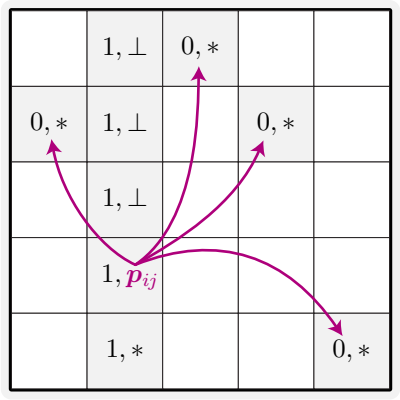


Figure 2 The unambiguous decision tree that defines the Tribes-List function.

columns. More formally, we define TL in Figure 2 by describing an unambiguous decision tree of cost $\Theta(k \log k)$ computing it.

3.2 Composition

Given a base function witnessing some complexity gap, we will establish a different but related complexity gap by transforming the function into a more complex one via one (or both) of the following operations involving function composition: *lifting* and AND-*composition*. Lifting is used to go from a query complexity gap to an analogous communication complexity gap. AND-composition is used to go from a gap with a UP upper bound to a gap with a 2UP upper bound. To show that an operation indeed converts one gap to another gap, we need two types of results: an observation showing how the relevant upper bounds behave under the operation, and a more difficult lemma showing how the relevant lower bounds behave under the operation.

Lifting

Let $g\colon \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ be a fixed two-party function (called the *gadget*). We can *lift* $f\colon \{0,1\}^n \to \{0,1\}$ via the gadget g to obtain a two-party composed function $f \circ g^n\colon (\{0,1\}^b)^n \times (\{0,1\}^b)^n \to \{0,1\}$ where Alice is given $x = (x_1, \ldots, x_n)$ and Bob is given $y = (y_1, \ldots, y_n)$ (with each $x_i, y_i \in \{0,1\}^b$) and the goal is to compute $(f \circ g^n)(x,y) \coloneqq f(g(x_1,y_1), \ldots, g(x_n,y_n))$.

A decision tree for f generally yields a corresponding type of communication protocol for $f \circ g^n$: whenever the decision tree queries the i-th bit, Alice and Bob communicate $b+1$ bits to evaluate the corresponding bit $g(x_i, y_i)$. By counting conjunctions, it can be verified that such a connection holds for the 2UP and UP models as well:

▶ **Observation 4.** *For all* $f\colon \{0,1\}^n \to \{0,1\}$, $g\colon \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$, *and* $\mathcal{C} \in \{\mathsf{2UP}, \mathsf{UP}\}$, *we have* $\mathcal{C}^{\mathsf{cc}}(f \circ g^n) \leq \mathcal{C}^{\mathsf{dt}}(f) \cdot O(b + \log n)$.

For any model $\mathcal{C}$, a result in the converse direction (giving a black-box method of converting a communication protocol for $f \circ g^n$ into a comparably efficient decision tree for

f) is highly nontrivial and is called a *simulation lemma*. In this work, we use a simulation lemma for $\mathcal{C} = \mathsf{WAPP}$:

▶ **Lemma 5** (Simulation for **WAPP** [15]). *For all* $f\colon \{0,1\}^n \to \{0,1\}$ *and constants* $0 < \epsilon < \delta < 1/2$, *we have* $\mathsf{WAPP}^{\mathsf{dt}}_{\delta}(f) \leq O\big(\mathsf{WAPP}^{\mathsf{cc}}_{\epsilon}(f \circ g^n)/\log n\big)$ *where* $g\colon \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ *is the inner-product gadget defined as follows:* $b = b(n) := 100 \log n$, *and* $g(x_i, y_i) := \langle x_i, y_i \rangle \bmod 2$.

AND-composition

Given $f\colon \{0,1\}^n \to \{0,1\}$ we can compose it with the k-bit AND function to obtain $\mathrm{AND} \circ f^k\colon (\{0,1\}^n)^k \to \{0,1\}$ defined by $(\mathrm{AND} \circ f^k)(z_1, \ldots, z_k) = 1$ iff $f(z_i) = 1$ for all i. Similarly, given $F\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ we can obtain $\mathrm{AND} \circ F^k\colon \mathcal{X}^k \times \mathcal{Y}^k \to \{0,1\}$ defined by $(\mathrm{AND} \circ F^k)(x, y) = 1$ iff $F(x_i, y_i) = 1$ for all i.

AND-composition converts a UP upper bound into a $\mathsf{2UP}$ upper bound [16]:

▶ **Observation 6.** *For all* f *and* k, *we have* $\mathsf{2UP}^{\mathsf{dt}}(\mathrm{AND} \circ f^k) \leq k \cdot \mathsf{UP}^{\mathsf{dt}}(f) + O(\mathsf{UP}^{\mathsf{dt}}(f)^2)$. *Similarly, for all* F *and* k, *we have* $\mathsf{2UP}^{\mathsf{cc}}(\mathrm{AND} \circ F^k) \leq k \cdot \mathsf{UP}^{\mathsf{cc}}(F) + O(\mathsf{UP}^{\mathsf{cc}}(F)^2 + \log k)$.

The two parts of Observation 6 are analogous, so we describe the idea only in terms of the query complexity part. Since $\mathsf{coUP}^{\mathsf{dt}}(f) \leq \mathsf{P}^{\mathsf{dt}}(f) \leq O(\mathsf{UP}^{\mathsf{dt}}(f)^2)$, it suffices to have $\mathsf{coUP}^{\mathsf{dt}}(f)$ as the second term on the right side. The idea is to let a 1-certificate for $\mathrm{AND} \circ f^k$ be comprised of 1-certificates for each of the k copies of f, and a 0-certificate for $\mathrm{AND} \circ f^k$ be comprised of a 0-certificate for the first copy of f that evaluates to 0, together with 1-certificates for each of the preceding copies of f.

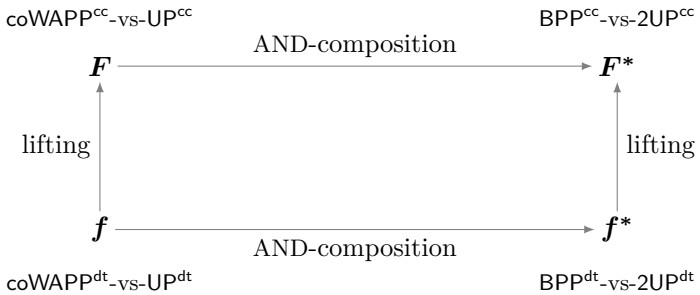
On the other hand, the following lemma (proven in Section 4.1) shows that randomized query complexity goes up by a factor of k under AND-composition.

▶ **Lemma 7.** *For all* f *and* k, *we have* $\mathsf{BPP}^{\mathsf{dt}}(f) \leq O\big(\mathsf{BPP}^{\mathsf{dt}}(\mathrm{AND} \circ f^k)/k\big)$.

We note that Lemma 7 qualitatively strengthens the tight direct sum result for randomized query complexity in [18] since computing the outputs of all k copies of f is at least as hard as computing the AND of the outputs. Similarly, if we could prove an analogue of Lemma 7 for communication complexity, it would qualitatively strengthen the notoriously-open tight direct sum conjecture for randomized communication complexity.

3.3 Overview of proofs

The following diagram shows how we construct the functions used to witness our gaps. Starting with some f, we can lift it to obtain F, or we can apply AND-composition to obtain f^*. We can obtain F^* by either lifting f^* or equivalently applying AND-composition to F.



Proof of Theorem 2

We start by discussing the proof of Theorem 2 as it will be used in the proof of Theorem 1. We actually prove the following stronger version of Theorem 2 that gives a lower bound even against $\mathsf{coWAPP}^{\mathsf{cc}}_{\epsilon}(F) \leq O(\mathsf{BPP}^{\mathsf{cc}}(F))$:

▶ **Theorem 2⋆ (coWAPP$^{\mathsf{cc}}$-vs-UP$^{\mathsf{cc}}$).** *There is an F s.t.* $\mathsf{coWAPP}^{\mathsf{cc}}_{0.04}(F) \geq \tilde{\Omega}(\mathsf{UP}^{\mathsf{cc}}(F)^2)$.

Our proof follows the same outline as in [16] and only requires us to lift the following analogous result for query complexity (proved in the full version [14]):

▶ **Lemma 8 (coWAPP$^{\mathsf{dt}}$-vs-UP$^{\mathsf{dt}}$).** $\mathsf{coWAPP}^{\mathsf{dt}}_{0.05}(\mathrm{TL}) \geq \tilde{\Omega}(\mathsf{UP}^{\mathsf{dt}}(\mathrm{TL})^2)$.

To derive Theorem 2⋆, set $f \coloneqq \mathrm{TL}$ and $F \coloneqq f \circ g^n$, where g is the gadget from Lemma 5 and n is the input length of f. Recall that $\mathsf{UP}^{\mathsf{dt}}(f) \geq n^{\Omega(1)}$. Thus by Observation 4, $\mathsf{UP}^{\mathsf{cc}}(F) \leq \mathsf{UP}^{\mathsf{dt}}(f) \cdot O(\log n) \leq \tilde{O}(\mathsf{UP}^{\mathsf{dt}}(f))$, and by Lemma 5, $\mathsf{coWAPP}^{\mathsf{cc}}_{0.04}(F) \geq \Omega(\mathsf{coWAPP}^{\mathsf{dt}}_{0.05}(f) \cdot \log n) \geq \Omega(\mathsf{coWAPP}^{\mathsf{dt}}_{0.05}(f))$. Thus $\mathsf{coWAPP}^{\mathsf{cc}}_{0.04}(F) \geq \tilde{\Omega}(\mathsf{UP}^{\mathsf{cc}}(F)^2)$.

Proof of Theorem 1

An "obvious" strategy for Theorem 1 would be again to first prove the analogous query complexity result and then lift it to communication complexity. (This is the outline used for the analogous result in [16].) In other words, we would follow the lower-right path in the above diagram:

Obvious strategy

(a) Start with f witnessing a $\mathsf{BPP}^{\mathsf{dt}}$-vs-$\mathsf{UP}^{\mathsf{dt}}$ gap.
(b) Obtain f^* witnessing a $\mathsf{BPP}^{\mathsf{dt}}$-vs-$2\mathsf{UP}^{\mathsf{dt}}$ gap by applying AND-composition to f.
(c) Obtain F^* witnessing a $\mathsf{BPP}^{\mathsf{cc}}$-vs-$2\mathsf{UP}^{\mathsf{cc}}$ gap by lifting f^*.

We have the tools to complete steps (a) and (b):

▶ **Lemma 9 (BPP$^{\mathsf{dt}}$-vs-2UP$^{\mathsf{dt}}$).** *There is an f such that* $\mathsf{BPP}^{\mathsf{dt}}(f) \geq \tilde{\Omega}(2\mathsf{UP}^{\mathsf{dt}}(f)^{1.5})$.

Proof. This is witnessed by $f^* \coloneqq \mathrm{AND} \circ \mathrm{TL}^k$ where $k \coloneqq \mathsf{UP}^{\mathsf{dt}}(\mathrm{TL})$. By Observation 6, $2\mathsf{UP}^{\mathsf{dt}}(f^*) \leq O(k^2)$, and by Lemmas 7–8,

$$\mathsf{BPP}^{\mathsf{dt}}(f^*) \geq \Omega(k \cdot \mathsf{BPP}^{\mathsf{dt}}(\mathrm{TL})) \geq \Omega(k \cdot \mathsf{coWAPP}^{\mathsf{dt}}_{0.05}(\mathrm{TL})) \geq \tilde{\Omega}(k^3)\,. \qquad \blacktriangleleft$$

Unfortunately, we do not know how to carry out step (c), because we currently lack a simulation lemma for BPP. (We believe that such a lemma is true, and it is an interesting open problem to prove this!) We get around this obstacle by reversing the order of steps (b) and (c), that is, we instead follow the upper-left path in the diagram:

Modified strategy

(a′) Start with f witnessing a $\mathsf{coWAPP}^{\mathsf{dt}}$-vs-$\mathsf{UP}^{\mathsf{dt}}$ gap.
(b′) Obtain F witnessing a $\mathsf{coWAPP}^{\mathsf{cc}}$-vs-$\mathsf{UP}^{\mathsf{cc}}$ gap by lifting f.
(c′) Obtain F^* witnessing a $\mathsf{BPP}^{\mathsf{cc}}$-vs-$2\mathsf{UP}^{\mathsf{cc}}$ gap by applying AND-composition to F.

Steps (a′) and (b′) are just Theorem 2⋆. For step (c′) it would suffice to have an analogue of Lemma 7 for communication complexity. This is open, but fortunately we have some wiggle room since it suffices to have coWAPP_ϵ instead of BPP on the left side of Lemma 7. For this, we *can* prove a communication analogue (indeed, with $\mathsf{2WAPP}_\epsilon$ instead of coWAPP_ϵ):

▶ **Lemma 10.** *For all F, k, and constants $0 < \epsilon < 1/2$, we have*

$$\mathsf{2WAPP}^{\mathsf{cc}}_\epsilon(F) \;\leq\; O\big(\mathsf{BPP}^{\mathsf{cc}}(\mathrm{AND} \circ F^k)/k + \log \mathsf{BPP}^{\mathsf{cc}}(\mathrm{AND} \circ F^k)\big).$$

To derive Theorem 1, let F be the function in Theorem 2⋆, and let $F^* := \mathrm{AND} \circ F^k$ where $k := \mathsf{UP}^{\mathsf{cc}}(F)$. Then F^* witnesses Theorem 1: By Observation 6, $\mathsf{2UP}^{\mathsf{cc}}(F^*) \leq O(k^2)$, and by Lemma 10, $\mathsf{BPP}^{\mathsf{cc}}(F^*) \geq \Omega\big(k \cdot (\mathsf{2WAPP}^{\mathsf{cc}}_{0.04}(F) - O(\log k))\big) \geq \Omega\big(k \cdot (\mathsf{coWAPP}^{\mathsf{cc}}_{0.04}(F) - O(\log k))\big) \geq \tilde{\Omega}(k^3)$.

Proof of Lemma 10

We start with the intuition for the proof of Lemma 7, which is a warmup for Lemma 10. For brevity let $f^* := \mathrm{AND} \circ f^k$. Given an input z for f, the basic idea is to plant z into a random coordinate of $f^*(z_1, \ldots, z_k)$, and plant random 1-inputs into the other coordinates, and then run the randomized decision tree for f^*. If q is the query complexity of f^*, the expected number of bits of z that are queried (over a random 1-input) will be at most q/k. Our new randomized decision tree will simulate this but abort after $8q/k$ queries to z have been made. If an answer is returned, we output the same value for $f(z)$, and if no answer is returned within this many queries, then we output 0. A simple analysis shows that we succeed with high probability in the average-case (which is equivalent to worst-case by the minimax theorem).

To prove Lemma 10, we would like to mimic this argument in the communication world, using the fact that internal information complexity is sandwiched between $\mathsf{BPP}^{\mathsf{cc}}$ and $\mathsf{2WAPP}^{\mathsf{cc}}$ [23] and satisfies a sort of AND-composition analogous to Lemma 7 using well-known properties (by planting the input into a random coordinate, and planting random 1-inputs into the other coordinates). However there is a significant barrier to this idea "just working": the AND-composition property (direct sum lemma) requires a distribution over 1-inputs of F (one-sided), while the relation to $\mathsf{2WAPP}^{\mathsf{cc}}$ requires an arbitrary distribution over inputs to F (two-sided). To bridge this divide, we prove a new property of information complexity: the one-sided version is essentially equivalent to the two-sided version. A key ingredient in showing the latter is the "information odometer" of [10], which allows us to keep track of the amount of information that has been revealed, and abort the protocol once we have reached our limit, and argue that we can carry this out without revealing too much extra information. We note that this one-vs-two sided information complexity lemma is the only component of the proof of Theorem 1 that distinguishes between arbitrary rectangle partitions ($\mathsf{2UP}^{\mathsf{cc}}$) and rectangle partitions induced by protocols (P^{cc}).

Organization

The only ingredients that remain to be proved are Lemma 8 (which we prove in the full version [14] and Lemma 7 and Lemma 10 (both of which we prove in Section 4).

4 AND-Composition Lemmas

In this section we prove Lemma 7 and Lemma 10, restated here for convenience.

▶ **Lemma 11.** *For all f and k, we have* $\mathsf{BPP}^{\mathsf{dt}}(f) \le O\big(\mathsf{BPP}^{\mathsf{dt}}(\mathrm{AND} \circ f^k)/k\big)$.

▶ **Lemma 12.** *For all F, k, and constants $0 < \epsilon < 1/2$, we have*

$$\mathsf{2WAPP}^{\mathsf{cc}}_{\epsilon}(F) \;\le\; O\big(\mathsf{BPP}^{\mathsf{cc}}(\mathrm{AND} \circ F^k)/k + \log \mathsf{BPP}^{\mathsf{cc}}(\mathrm{AND} \circ F^k)\big).$$

4.1 AND-composition for query complexity

We now prove Lemma 7. For brevity let $f^* \coloneqq \mathrm{AND} \circ f^k$. Let T^* be a height-q randomized decision tree for f^* with error $1/8$. We design a height-$8q/k$ randomized decision tree for f with error $1/4$.

Let D be an arbitrary distribution over $f^{-1}(1)$. Consider the following randomized decision tree T that takes $z \in \{0,1\}^n$ as input:

1. Pick $i \in [k]$ uniformly at random and let $z_i \coloneqq z$.
2. For $j \in [k] \setminus \{i\}$ sample $z_j \sim D$ independently.
3. Run $T^*(z_1, \ldots, z_k)$ until it has made $8q/k$ queries in the i-th component.
4. If T^* already produced an output in Step 3, output the same bit; else output 0.

Note that with probability 1 we have $f^*(z_1, \ldots, z_k) = f(z)$. Let R_T denote T's randomness and R_{T^*} denote T^*'s randomness. If $f(z) = 0$ then

$$\mathbb{P}_{R_T}[T(z) = 1] \;\le\; \max_{(z_1,\ldots,z_k) \in (f^*)^{-1}(0)} \mathbb{P}_{R_{T^*}}[T^*(z_1, \ldots, z_k) = 1] \;\le\; 1/8 \;\le\; 1/4.$$

Furthermore,

$$\begin{aligned}
\mathbb{P}_{z\sim D,\, R_T}[T(z) = 0] \;&=\; \mathbb{P}_{z_1,\ldots,z_k \sim D,\, i\in[k],\, R_{T^*}}\begin{bmatrix} T^*(z_1,\ldots,z_k) \text{ outputs 0 or makes more} \\ \text{than } 8q/k \text{ queries in the } i\text{-th component} \end{bmatrix} \\
&\le\; \max_{(z_1,\ldots,z_k)\in(f^*)^{-1}(1)} \left(\begin{array}{l} \mathbb{P}_{R_{T^*}}[T^*(z_1,\ldots,z_k) = 0] + \\ \max_{R_{T^*}} \mathbb{P}_{i\in[k]}\begin{bmatrix} T^*(z_1,\ldots,z_k) \text{ makes more than} \\ 8q/k \text{ queries in the } i\text{-th component} \end{bmatrix} \end{array} \right) \\
&\le\; 1/8 + 1/8 \;=\; 1/4.
\end{aligned}$$

Now let D be an arbitrary distribution over $\{0,1\}^n$ and define T w.r.t. $(D \,|\, f^{-1}(1))$. We have

$$\begin{aligned}
\mathbb{P}_{z\sim D,\, R_T}[T(z) \neq f(z)] \;&=\; \textstyle\sum_{b\in\{0,1\}} \mathbb{P}_{z\sim(D\,|\,f^{-1}(b)),\, R_T}[T(z) \neq b] \cdot \mathbb{P}_{z\sim D}[f(z) = b] \\
&\le\; \textstyle\sum_{b\in\{0,1\}} (1/4) \cdot \mathbb{P}_{z\sim D}[f(z) = b] \;=\; 1/4.
\end{aligned}$$

By the minimax theorem, there is a height-$8q/k$ randomized decision tree (a mixture of the T's) that on any input produces the wrong output with probability $\le 1/4$.

4.2 Definitions

We adopt the following conventions throughout the proof of Lemma 10. We denote random variables with upper-case letters, and we denote particular outcomes of the random variables with the corresponding lower-case letters. All communication protocols are randomized and mixed-coin, and we use (R, R_A, R_B) to denote the public randomness, Alice's private randomness, and Bob's private randomness, respectively. We say a protocol Π is ϵ-correct for F if for all (x, y), $\mathbb{P}_{R,R_A,R_B}[\Pi(x, y) = F(x, y)] \ge 1 - \epsilon$. For a distribution D over

inputs, we say Π is (ϵ, D)-correct for F if $\mathbb{P}_{(X,Y)\sim D,\, R,R_A,R_B}[\Pi(X,Y) = F(X,Y)] \geq 1-\epsilon$. The internal information cost of a protocol Π with respect to $(X,Y) \sim D$ is defined as $\mathrm{IC}_D(\Pi) := \mathbb{I}(R,M\,;X\,|\,Y) + \mathbb{I}(R,M\,;Y\,|\,X) = \mathbb{I}(M\,;X\,|\,Y,R) + \mathbb{I}(M\,;Y\,|\,X,R)$ where the random variable M is the concatenation of all messages. We also let $\mathrm{CC}(\Pi)$ denote the worst-case communication cost of Π.

It is convenient for us to work with a measure $\mathsf{2WAPP}^{\mathsf{cc}*}$ that is defined slightly differently from $\mathsf{2WAPP}^{\mathsf{cc}}$ but is equivalent in the sense that for all F and $0 < \epsilon < 1/2$, $\mathsf{2WAPP}^{\mathsf{cc}}_{\epsilon}(F) \leq \mathsf{2WAPP}^{\mathsf{cc}*}_{\epsilon}(F) \leq O(\mathsf{2WAPP}^{\mathsf{cc}}_{\epsilon/2}(F))$. We note that $\mathsf{2WAPP}^{\mathsf{cc}}$ directly expresses the two-sided smooth rectangle bound of [17], while $\mathsf{2WAPP}^{\mathsf{cc}*}$ directly expresses the relaxed partition bound of [23] and was the definition used in [15].

▶ **Definition 13.** We define $\mathsf{2WAPP}^{\mathsf{cc}*}_{\epsilon}(F)$ as the minimum of $\mathrm{CC}(\Pi)+\log(1/\alpha)$ over all $\alpha > 0$ and all protocols Π with output values $\{0,1,\perp\}$ such that for all (x,y), $\mathbb{P}[\Pi(x,y) \neq \perp] \leq \alpha$ and $\mathbb{P}[\Pi(x,y) = F(x,y)] \geq (1-\epsilon)\alpha$ (i.e., Π is $(1-(1-\epsilon)\alpha)$-correct).

We also need the distributional version of $\mathsf{2WAPP}^{\mathsf{cc}*}$.

▶ **Definition 14.** For an input distribution D, we define $\mathsf{2WAPP}^{\mathsf{cc}*}_{\epsilon,D}(F)$ as the minimum of $\mathrm{CC}(\Pi) + \log(1/\alpha)$ over all $\alpha > 0$ and all protocols Π with output values $\{0,1,\perp\}$ such that $\mathbb{P}[\Pi(x,y) \neq \perp] \leq \alpha$ for all (x,y), and $\mathbb{P}[\Pi(X,Y) = F(X,Y)] \geq (1-\epsilon)\alpha$ for $(X,Y) \sim D$ (i.e., Π is $(1-(1-\epsilon)\alpha, D)$-correct).

4.3 AND-composition for communication complexity

We now outline the proof of Lemma 10. Recall that the proof of Lemma 7 involved these steps:

- **(i)** embedding the input into a random coordinate of a k-tuple and filling the other coordinates with random 1-inputs (to cut the cost on 1-inputs by a factor k),
- **(ii)** aborting the execution if the cost became too high (to ensure low cost also on 0-inputs while maintaining average-case correctness on 1-inputs),
- **(iii)** using the minimax theorem to go from average-case to worst-case correctness.

We start by noting that an analogue of (i) holds for information complexity (which lower bounds $\mathsf{BPP}^{\mathsf{cc}}$). Then as one of our main technical contributions we prove an analogue of (ii) for information complexity. Then inbetween (ii) and (iii) we insert a step applying the known result that information complexity upper bounds $\mathsf{2WAPP}^{\mathsf{cc}*}$ in the distributional setting. Finally we use the analogue of (iii) for $\mathsf{2WAPP}^{\mathsf{cc}*}$. Formally, Lemma 10 follows by stringing together the following lemmas.

▶ **Lemma 15.** *Fix any F, k, $0 < \epsilon < 1/2$, and distribution D over $F^{-1}(1)$. For every ϵ-correct protocol Π for $\mathrm{AND} \circ F^k$ there is an ϵ-correct protocol Π' for F with $\mathrm{IC}_D(\Pi') \leq \mathrm{CC}(\Pi)/k$ and $\mathrm{CC}(\Pi') \leq \mathrm{CC}(\Pi)$.*

▶ **Lemma 16.** *Fix any F, constants $0 < \epsilon < \delta < 1/2$, and input distribution D, and let $D^1 := (D\,|\,F^{-1}(1))$. For every (ϵ, D)-correct protocol Π there is a (δ, D)-correct protocol Π' with $\mathrm{IC}_D(\Pi') \leq O\big(\mathrm{IC}_{D^1}(\Pi) + \log(\mathrm{CC}(\Pi)+2)\big)$.*

▶ **Lemma 17.** *Fix any F, constants $0 < \epsilon < \delta < 1/2$, and input distribution D. For every (ϵ, D)-correct protocol Π we have $\mathsf{2WAPP}^{\mathsf{cc}*}_{\delta,D}(F) \leq O(\mathrm{IC}_D(\Pi) + 1)$.*

▶ **Lemma 18.** *Fix any F and $0 < \epsilon < 1/2$. Then $\mathsf{2WAPP}^{\mathsf{cc}*}_{\epsilon}(F) \leq 2 + \max_D \mathsf{2WAPP}^{\mathsf{cc}*}_{\epsilon,D}(F)$.*

Lemma 15 is a standard application of the "direct sum" property of information cost. Lemma 16 is proved in Section 4.4 and relies on [10]. Lemma 17 is due to [23, Theorem 1.1 of the ECCC version]. Lemma 18 follows from an argument in [23, Appendix A of the ECCC version] that uses LP duality.

The moral conclusion of Lemma 16 is that "one-sided information complexity" is essentially equivalent to "two-sided information complexity" for average-case protocols. Combining Lemma 16 with [9, Theorem 3.5 of the ECCC version] shows that a similar equivalence holds for worst-case protocols. More specifically, a distribution-independent definition of information complexity for bounded-error protocols can be obtained by maximizing over all input distributions; our corollary shows that this measure is essentially unchanged if we maximize only over distributions over 1-inputs (or symmetrically, 0-inputs).

▶ **Corollary 19.** *Fix any F, constants $0 < \epsilon < \delta < 1/2$, and $b \in \{0,1\}$. Then*

$$\inf_{\substack{\delta\text{-correct} \\ \text{protocols } \Pi}} \max_{\substack{D \text{ over} \\ \text{all inputs}}} \mathrm{IC}_D(\Pi) \;\leq\; \max_{\substack{D \text{ over} \\ b\text{-inputs}}} \inf_{\substack{\epsilon\text{-correct} \\ \text{protocols } \Pi}} O\big(\mathrm{IC}_D(\Pi) + \log(\mathrm{CC}(\Pi) + 2)\big).$$

Theorem 3 follows by swapping the quantifiers on the right side of the inequality in Corollary 19 (which only weakens the statement), and by straightforwardly accounting for the communication cost in the proof. We can also assume the protocol Π' has error $\leq 1/3$ by a standard error reduction technique (take a majority vote of several runs of the protocol), which does not affect information complexity except by constant factors. We do not directly employ this worst-case version of Lemma 16, but it is used in the follow-up work [5].

4.4 One-sided information vs. two-sided information

Intuition for Lemma 16

Recall the following idea, which was implicit in the proof of Lemma 7. Suppose we have a randomized decision tree computing some function, and we have a bound b on the expected number of queries made over a random 1-input. Then to obtain a randomized decision tree with a worst-case query bound, we can keep track of the number of queries made during the execution and halt and output 0 if it exceeds, say, $8b$. Correctness on 0-inputs is maintained since we either run the original decision tree to completion and thus output 0 with high probability, or we abort and output 0 anyway. We get average-case correctness on 1-inputs since by Markov's inequality, with probability at least $7/8$ the original decision tree uses at most $8b$ queries, in which case we run it to completion and output 1 with high probability.

The high-level intuition is to mimic this idea for information complexity. We have a protocol with a bound on the information cost w.r.t. the distribution D^1 over 1-inputs. The "information odometer" of [10] allows us to "keep track of" information cost, so we can halt and output 0 if it becomes too large. This will guarantee that the information cost is low w.r.t. the input distribution D, and correctness on 0-inputs is maintained. However, there is a complication with showing the average-case correctness on 1-inputs.

For each computation path specified by an input (x, y), an outcome of public randomness r, and a full sequence of messages m, there is a contribution $c_{x,y,r,m}$ such that the information cost w.r.t. D is the expectation of $c_{x,y,r,m}$ over a random computation path with $(x, y) \sim D$. Similarly, there is a contribution $c^1_{x,y,r,m}$ such that the information cost w.r.t. D^1 is the expectation of $c^1_{x,y,r,m}$ over a random computation path with $(x, y) \sim D^1$. These contributions play the role of "number of queries" along a computation path in the decision tree setting, but a crucial difference is that $c_{x,y,r,m} \neq c^1_{x,y,r,m}$ in general; i.e., the contribution to information cost depends on the input distribution (whereas number of queries did not). To show the

average-case correctness on 1-inputs, we need a bound on the typical value of $c_{x,y,r,m}$, whereas the assumption that information cost w.r.t. D^1 is low gives us a bound on the typical value of $c^1_{x,y,r,m}$.

Thus the heart of the argument is to show that typically, $c_{x,y,r,m}$ is not much larger than $c^1_{x,y,r,m}$. Intuitively, one might expect the difference to be at most 1, since the only additional information that can be revealed (beyond what is revealed under D^1) should be the fact that (x, y) is a 1-input (which is 1 bit of information). More precisely, we show that for given (x, y), the expected difference depends on how balanced F is on the x row and the y column. Then we just need to note that F is typically reasonably balanced for both the x row and the y column.

The formal proof of Lemma 16 is deferred to the full version [14] due to space constraints.

Acknowledgments. We thank Mark Braverman, Troy Lee, and Omri Weinstein for discussions. Work done by M.G. while at IBM Research Almaden.

References

1 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the 48th Symposium on Theory of Computing (STOC)*, pages 863–876. ACM, 2016. doi:10.1145/2897518.2897644.

2 Alfred Aho, Jeffrey Ullman, and Mihalis Yannakakis. On notions of information transfer in VLSI circuits. In *Proceedings of the 15th Symposium on Theory of Computing (STOC)*, pages 133–139. ACM, 1983. doi:10.1145/800061.808742.

3 Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. In *Proceedings of the 48th Symposium on Theory of Computing (STOC)*, pages 800–813. ACM, 2016. doi:10.1145/2897518.2897524.

4 Andris Ambainis, Martins Kokainis, and Robin Kothari. Nearly optimal separations between communication (or query) complexity and partitions. In *Proceedings of the 31st Computational Complexity Conference (CCC)*, pages 4:1–4:14. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.CCC.2016.4.

5 Anurag Anshu, Aleksandrs Belovs, Shalev Ben-David, Mika Göös, Rahul Jain, Robin Kothari, Troy Lee, and Miklos Santha. Separations in communication complexity using cheat sheets and information complexity. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 555–564. IEEE, 2016. doi:10.1109/FOCS.2016.66.

6 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004. doi:10.1016/j.jcss.2003.11.006.

7 Aleksandrs Belovs. Non-intersecting complexity. In *Proceedings of the 32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 158–165. Springer, 2006. doi:10.1007/11611257_13.

8 Elmar Böhler, Christian Glaßer, and Daniel Meister. Error-bounded probabilistic computations between MA and AM. *Journal of Computer and System Sciences*, 72(6):1043–1076, 2006. doi:10.1016/j.jcss.2006.05.001.

9 Mark Braverman. Interactive information complexity. *SIAM Journal on Computing*, 44(6):1698–1739, 2015. doi:10.1137/130938517.

10 Mark Braverman and Omri Weinstein. An interactive information odometer and applications. In *Proceedings of the 47th Symposium on Theory of Computing (STOC)*, pages 341–350. ACM, 2015. doi:10.1145/2746539.2746548.

11 Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *Journal of the ACM*, 62(2):17:1–17:23, 2015. doi:10.1145/2716307.

12 Dmitry Gavinsky and Shachar Lovett. En Route to the Log-Rank Conjecture: New Reductions and Equivalent Formulations. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 514–524. Springer, 2014. doi:10.1007/978-3-662-43948-7_43.

13 Mika Göös. Lower bounds for clique vs. independent set. In *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS)*, pages 1066–1076. IEEE, 2015. doi:10.1109/FOCS.2015.69.

14 Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication vs. partition number. Technical Report TR15-169, Electronic Colloquium on Computational Complexity (ECCC), 2015. URL: https://eccc.weizmann.ac.il/report/2015/169/.

15 Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016. doi:10.1137/15M103145X.

16 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS)*, pages 1077–1088. IEEE, 2015. doi:10.1109/FOCS.2015.70.

17 Rahul Jain and Hartmut Klauck. The partition bound for classical communication complexity and query complexity. In *Proceedings of the 25th Conference on Computational Complexity (CCC)*, pages 247–258. IEEE, 2010. doi:10.1109/CCC.2010.31.

18 Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Information Processing Letters*, 110(20):893–897, 2010. doi:10.1016/j.ipl.2010.07.020.

19 Rahul Jain, Troy Lee, and Nisheeth Vishnoi. A quadratically tight partition bound for classical communication complexity and query complexity. Technical report, arXiv, 2014. arXiv:1401.4512.

20 T.S. Jayram, Swastik Kopparty, and Prasad Raghavendra. On the communication complexity of read-once AC^0 formulae. In *Proceedings of the 24th Conference on Computational Complexity (CCC)*, pages 329–340. IEEE, 2009. doi:10.1109/CCC.2009.39.

21 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.

22 Jedrzej Kaniewski, Troy Lee, and Ronald de Wolf. Query complexity in expectation. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 761–772. Springer, 2015. doi:10.1007/978-3-662-47672-7_62.

23 Iordanis Kerenidis, Sophie Laplante, Virginie Lerays, Jérémie Roland, and David Xiao. Lower bounds on information complexity via zero-communication protocols and applications. *SIAM Journal on Computing*, 44(5):1550–1572, 2015. doi:10.1137/130928273.

24 Gillat Kol, Shay Moran, Amir Shpilka, and Amir Yehudayoff. Approximate nonnegative rank is equivalent to the smooth rectangle bound. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 701–712. Springer, 2014. doi:10.1007/978-3-662-43948-7_58.

25 Robin Kothari, David Racicot-Desloges, and Miklos Santha. Separating decision tree complexity from subcube partition complexity. In *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, pages 915–930. Schloss Dagstuhl, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.915.

26 Eyal Kushilevitz. Unpublished. Cited in [32], 1994.

27 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

28 Troy Lee and Adi Shraibman. Lower bounds in communication complexity. *Foundations and Trends in Theoretical Computer Science*, 3(4):263–399, 2007. doi:10.1561/0400000040.

29 Nikos Leonardos and Michael Saks. Lower bounds on the randomized communication complexity of read-once functions. *Computational Complexity*, 19(2):153–181, 2010. doi:10.1007/s00037-010-0292-2.

30 László Lovász and Michael Saks. Lattices, Möbius functions and communication complexity. In *Proceedings of the 29th Symposium on Foundations of Computer Science (FOCS)*, pages 81–90. IEEE, 1988. doi:10.1109/SFCS.1988.21924.

31 Sagnik Mukhopadhyay and Swagato Sanyal. Towards better separation between deterministic and randomized query complexity. In *Proceedings of 35th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 206–220. Schloss Dagstuhl, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.206.

32 Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995. doi:10.1007/BF01192527.

33 Petr Savický. On determinism versus unambiguous nondeterminism for decision trees. Technical Report TR02-009, Electronic Colloquium on Computational Complexity (ECCC), 2002. URL: http://eccc.hpi-web.de/report/2002/009/.

34 Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991. doi:10.1016/0022-0000(91)90024-Y.

Approximate Bounded Indistinguishability

Andrej Bogdanov[1] **and Christopher Williamson**[2]

1 **Department of Computer Science and Engineering and Institute for Theoretical Computer Science and Communications, Chinese University of Hong Kong, Hong Kong, China**
andrejb@cse.cuhk.edu.hk

2 **Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China**
chris@cse.cuhk.edu.hk

Abstract

Two distributions over n-bit strings are (k, δ)-wise indistinguishable if no statistical test that observes k of the n bits can tell the two distributions apart with advantage better than δ. Motivated by secret sharing and cryptographic leakage resilience, we study the existence of pairs of distributions that are (k, δ)-wise indistinguishable, but can be distinguished by some function f of suitably low complexity. We prove bounds tight up to constants when f is the OR function, and tight up to logarithmic factors when f is a read-once uniform AND $\circ$ OR formula, extending previous works that address the perfect indistinguishability case $\delta = 0$.

We also give an elementary proof of the following result in approximation theory: If p is a univariate degree-k polynomial such that $|p(x)| \leq 1$ for all $|x| \leq 1$ and $p(1) = 1$, then $\hat{\ell}_1(p) \geq 2^{\Omega(p'(1)/k)}$, where $\hat{\ell}_1(p)$ is the sum of the absolute values of p's coefficients. A more general statement was proved by Servedio, Tan, and Thaler (2012) using complex-analytic methods.

As a secondary contribution, we derive new threshold weight lower bounds for bounded depth AND-OR formulas.

1998 ACM Subject Classification F.0 [Theory of Computation] General

Keywords and phrases pseudorandomness, polynomial approximation, secret sharing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.53

1 Introduction

Two random variables X and Y over $\{0,1\}^n$ are (locally) (k, δ)-*wise indistinguishable* if for all subsets $S \subseteq \{1, \ldots, n\}$ of size k, the induced marginal distributions $(X_i : i \in S)$ and $(Y_i : i \in S)$ are within statistical distance δ. We say function $f : \{0,1\}^n \to \{0,1\}$ *reconstructs from* the pair (X, Y) with error at most ε if $\mathbb{E}[f(X)] - \mathbb{E}[f(Y)] \geq 1 - \varepsilon$. In this work we investigate conditions under which a suitable f can reconstruct from some pair of locally indistinguishable distributions.

The parity function on n bits provides an extreme example of this phenomenon: The uniform distributions over $\{0,1\}^n$ conditioned on the parity of all the bits taking value zero and one, respectively, are $(n-1, 0)$-wise indistinguishable, but parity reconstructs from them perfectly. Our focus will be on reconstruction functions that have representations of constant depth and size polynomial in n (i.e., in the class AC^0). Functions in this class exclude large parities [6, 18, 7], and are in fact strongly uncorrelated with them under the uniform distribution [8].

The OR function

The OR function on n inputs is arguably the most elementary example of this type. We prove matching upper and lower bounds reconstruction by the OR function with respect to (k, δ)-wise indistinguishable distributions. In the range where the reconstruction error is constant, we obtain the following results.

▶ **Theorem 1.** *For every n and $k \geq 5\sqrt{n}$ and every pair of $(k, 2^{-O(n/k)})$-wise indistinguishable distributions X, Y over $\{0,1\}^n$, $\Pr[\mathrm{OR}(X) = 1] - \Pr[\mathrm{OR}(Y) = 1] \leq 1/3$.*

▶ **Theorem 2.** *For every n and $k \leq n/2$ there exists a pair of $(k, 2^{-\Omega(n/k)})$-wise indistinguishable distributions X, Y over $\{0,1\}^n$ such that $\Pr[\mathrm{OR}(X) = 1] - \Pr[\mathrm{OR}(Y) = 1] \geq 2/3$.*

The distributions X and Y in Theorem 2 are uniformly sampleable by circuit families of constant depth and size polynomial in n.

These results extend our previous work with Ishai and Viola [3], which only considered perfect bounded indistinguishability, i.e., the case $\delta = 0$. It is shown there that $(2\sqrt{n}, 0)$-wise and $(\sqrt{n}/2, 0)$-wise indistinguishability yield the conclusions of Theorems 1 and 2, respectively. Those proofs make use of results about the approximability of the OR function by real-valued polynomials of Linial and Nisan [10] and Nisan and Szegedy [13].

Our work with Ishai and Viola also explains the relevance of bounded indistinguishability to the computational complexity of secret sharing and cryptographic leakage resilience. In the context of secret sharing, (k, δ)-wise indistinguishability postulates that the joint view of any k parties can predict the secret with advantage at most δ. In a *visual secret sharing* scheme [12], the secret to be shared is a pixel, each of n parties receives a transparency and the pixel is recovered by superimposing the transparencies. The procedure can be applied independently to every pixel in an image. The *contrast* of the scheme is the fraction of pixels that are reconstructed correctly.

Theorems 1 and 2 describe the best possible tradeoff between the size and the reconstruction advantage of the adversarial coalition for visual secret sharing schemes [12] with constant contrast. Theorem 2' in Section 3 is a refinement of Theorem 2 that specifies the dependence of the indistinguishability parameters on the contrast.

Linial and Nisan [10] studied an incomparable notion of δ-indistinguishability in which the family of statistical tests consists of ANDs over arbitrary subsets of the input bits. They proved analogues of Theorems 1 and 2 for $\delta \geq 2^{-c\sqrt{n}}$ and $\delta \leq 2^{-c\sqrt{n}\log n}$, respectively (for suitable constants c and C).

A consequence in approximation theory

A. A. Markov showed that among all real-valued univariate degree k polynomials p such that $|p(x)| \leq 1$ for all $|x| \leq 1$, the derivative $p'(1)$ is maximized by the Chebyshev polynomial T_k of degree k with $T_k'(1) = k^2$. The weight $\hat{\ell}_1(T_k)$ of the Chebyshev polynomial, defined as the sum of the absolute values of its coefficients, is exponential in k. At the other end of the spectrum, the polynomial x^k has weight 1 and derivative k at 1. Interpolating between the two, the degree-k polynomial $T_r(x^{k/r})$ has weight exponential in r and derivative rk at 1 whenever r divides k. With one small additional hypothesis, we prove that this is the best possible up to the constant in the exponent:

▶ **Theorem 3.** *There exists a constant $C > 0$ such that if $p\colon \mathbb{R} \to \mathbb{R}$ is a degree-k polynomial with $|p(x)| \leq 1$ for all $|x| \leq 1$ and $p(1) = 1$ then $p'(1) \leq Ck \log \hat{\ell}_1(p)$.*

The polynomials $p(x) = T_r(x^{\lfloor k/r \rfloor})$ certify that the bound is tight up to the constant factor.

Servedio, Tan, and Thaler [14] proved a more general form of Theorem 3: Under the same assumptions, they showed that $\max_{x\in[-1,1]}|p'(x)| \leq Ck \log \hat{\ell}_1(p)$. (Their bound is stated in a slightly weaker form.) Our proof is based on elementary counting, a large deviation bound, and some basic calculus, while Servedio et al.'s makes use of Hadamard's Three Circle Theorem from complex analysis.

Perfect and almost-perfect reconstruction

In the setting of secret sharing, reconstruction errors are undesirable as they entail possible loss of information in the sharing phase. There, perfect reconstruction is a desirable feature. The OR function is not up to the task (for non-trivial parameter settings): It is easily observed that perfect reconstruction by OR requires $(1, 1/n)$-wise indistinguishability of the underlying distributions X and Y. We show, however, that read-once CNFs and higher depth AND-OR trees can perfectly reconstruct from distributions of approximate bounded indistinguishability:

▶ **Theorem 4.** *For any fixed d, and for all n and k, there exists a pair of $(k, 2^{-\Omega((n/k)^{1-1/d})})$-wise indistinguishable distributions that can be perfectly reconstructed by the depth-d AND-OR tree with top fan-in $(n/k)^{1/d}$, middle fan-ins $(n/k)^{2/d}$, and bottom fan-in $\frac{k^{(2d-3)/d}}{n^{(d-3)/d}}$.*

Setting $d = 2$ results in the following corollary:

▶ **Corollary 5.** *For all n and k there exists a pair of $(k, 2^{-\Omega((n/k)^{1/2})})$-wise indistinguishable distributions that can be perfectly reconstructed by the function $\mathrm{AND}_{(n/k)^{1/2}} \circ \mathrm{OR}_{(nk)^{1/2}}$.*

Here, $\mathrm{AND}_{n/m} \circ \mathrm{OR}_m$ is a read-once monotone CNF with fan-in m at the bottom OR gates and fan-in n/m at the top AND gate. We prove that Corollary 5 is essentially tight for read-once CNFs. In Proposition 13, however, we show that *almost* perfect reconstruction by functions of this type is possible with substantially better parameters.

Threshold weight

The degree-k threshold weight of a function $f\colon \{0,1\}^n \to \{0,1\}$ is the minimum weight of a nonzero degree-k polynomial p with integer coefficients such that $p(x)f(x) \geq 0$ for all x. Beigel [1] and Servedio et al. [14] construct a length-n decision list that requires degree-k threshold weight $2^{\Omega(\sqrt{n/k})}$. Bun and Thaler [5] give a read-once DNF over n variables that requires the same degree-k threshold weight, and construct a polynomial size AND-OR circuit of depth 3 that requires degree-k threshold weight $2^{\Omega(n/k^{3/2})}$. Sherstov [16] constructed a depth 4 circuit of polynomial size that requires threshold weight $2^{\Omega(\sqrt{n})}$ for all k. Our methods yield the following incomparable bound:

▶ **Corollary 6.** *The depth-d AND-OR tree given in the statement of Theorem 4 requires degree-k threshold weight $2^{\Omega((n/k)^{1-1/d})}$.*

In Section 6 we show that this result implies the degree-independent threshold weight lower bound for formulas of Sherstov [15].

Our proofs

Impossibility of reconstruction by OR from $(2\sqrt{n}, 0)$-wise indistinguishable distributions follows easily from the existence of a degree $2\sqrt{n}$ polynomial p that approximates the OR

function pointwise: The distinguishing advantage of the OR function can be at most the pointwise approximation error. The relaxed assumption of approximate local indistinguishability introduces an additional error term proportional to the weight of the approximating polynomial. Lemma 7 recasts the problem in the univariate setting, following previous works. To prove Theorem 1 we instantiate Lemma 7 with a polynomial of the form $T_r(x^{k/r})$ for a suitable choice of r. The relevant properties of this polynomial are that it is bounded on $[-1, 1]$, has weight $2^{O(r)}$ and has value $\Omega(kr/n)$ at $x = 1 + 1/2n$.

In the setting of perfect bounded indistinguishability, the maximum distinguishing advantage of OR with respect to $(k, 0)$-wise indistinguishable distributions *equals* the minimum error that a k-approximating polynomial for OR must have by linear programming duality. Thus, high approximate degree readily implies the existence of locally indistinguishable distributions that can be told apart by the OR function. In the approximate setting this duality is not preserved: While existence of low-degree, low-weight approximate polynomials implies hardness of reconstruction, it is not at all clear that the converse should hold. To prove a converse to Theorem 1 we instead resort to combinatorial means.

We prove Theorem 2 by a reduction to the case of perfect bounded indistinguishability. Previous works give the existence of $(\Omega(\sqrt{\varepsilon n}), 0)$-wise indistinguishable distributions X and Y such that $\Pr[\mathrm{OR}(X) = 1] - \Pr[\mathrm{OR}(Y) = 1] \geq 1 - \varepsilon$. (As observed in [3], the dual polynomials of Špalek [17] and Bun and Thaler [4] show that such distributions can be sampled by uniform constant-depth, polynomial-size circuit families.) We consider the following distributions X' and Y' over $\{0, 1\}^N$, where $N \geq n$: Sample n coordinates of $\{1, \ldots, N\}$ uniformly at random, embed a sample of X and Y, respectively, in these coordinates, and set all the other entries of X and Y to zero. The distinguishing advantage of OR is not affected by this transformation. On the other hand, any "local view" of size K in $\{1, \ldots, N\}$ expects to observe $K \cdot n/N$ of the embedded samples. Provided this number is sufficiently smaller than the indistinguishability parameter k, by a large deviation bound this local view can reconstruct from the distributions X and Y only with small probability.

Since Theorem 2 proves the optimality of Theorem 1, it follows that no choice of univariate polynomial p that is bounded on $[-1, 1]$ can have significantly larger value at $x = 1 + 1/2n$ than the polynomial $T_r(x^{k/r})$ among all those of weight $2^{O(r)}$. By the mean value theorem, there must exist some x in $[1, 1 + 1/2n]$ such that $p'(x)$ is at most $O(kr)$. Proving Theorem 3 requires showing that $p'(1) = O(kr)$. Our proof of Theorem 3 in Section 4 bounds the rate of change of $p'(x)$ around $x = 1$ as a function of the weight of p. Since this norm is small, for a suitable choice of n we can conclude from a somewhat delicate calculation that $p'(1) = O(kr)$ and prove Theorem 3.

The proof of Theorem 4 extends the reduction to perfect bounded indistinguishability to the setting of higher depth trees. This was studied in [3] and is a consequence of the seminal lower bound of Minsky and Papert [11]. Optimality is proved by an explicit construction of low-weight approximating polynomials as in Theorem 1. In the near-perfect setting, the requisite lower bound on approximate degree was obtained by Bun and Thaler [5] (extending Beigel [2]).

2 Optimality of distributions: Proof of Theorem 1

We first present a limitation of the OR function's ability to reconstruct from distributions that are almost k-wise indistinguishable. This is completed in two steps. First, we reduce the question to one of polynomials. Specifically, we demonstrate that the existence of certain polynomials that approximate OR allow us to upper bound the ability of OR to reconstruct

from these distributions, where the degree and size of coefficients of the polynomial relate to the indistinguishability parameters. The second step is to construct these polynomials and calculate the indistinguishability parameters as a function of their degree and $\hat{\ell}_1$ norm.

2.1 Indistinguishability from approximating polynomials

▶ **Lemma 7.** *Assume $p\colon \mathbb{R} \to \mathbb{R}$ is a degree-k polynomial such that $|p(x)| \le 1$ for all x such that $|x| \le 1$ and $j(p,n) = p(1+2/n) - 1 \ge 0$. For all pairs of (k,δ)-wise indistinguishable distributions X and Y over $\{-1,1\}^n$,*

$$\mathbb{E}[\mathrm{OR}(X)] - \mathbb{E}[\mathrm{OR}(Y)] \le \frac{2}{2 + j(p,n)} + e^2 \cdot \delta \cdot \hat{\ell}_1(p).$$

Here, we assume a representation in which OR evaluates to zero if any of its inputs are -1 and to 1 otherwise.

In the case of perfect indistinguishability (i.e., $\delta = 0$), such approximations of the OR function by polynomials already appear in the work of Linial and Nisan. Lemma 7 shows that the presence of local error δ introduces an additional term proportional to the weight of the approximation polynomial. The term $j(p,n)$ can be improved slightly to $p(1+2/(n-1)) - 1$.

In the proof of Lemma 7 we will use the following facts, which themselves are proven after the lemma.

▶ **Fact 8.** *Let p be a degree-k univariate polynomial and $p^*(x) = p(ax+b)$, where $|a|+|b| \ge 1$. Then $\hat{\ell}_1(p^*) \le (|a| + |b|)^k \hat{\ell}_1(p)$.*

▶ **Fact 9.** *If p is a univariate polynomial and $\mathbf{p}(x_1, \ldots, x_n) = p((x_1 + \cdots + x_n)/n)$ then $\hat{\ell}_1(\mathbf{p}) \le \hat{\ell}_1(p)$.*

Proof of Lemma 7. Let

$$p^*(x) = \gamma \cdot p\Big(x + \frac{2}{n}\Big) \quad \text{where} \quad \gamma = \frac{1}{2 + j(p,n)}.$$

Then $|p^*(x)| \le \gamma$ for $x \in [-1, 1 - \frac{2}{n}]$ and $p^*(1) = 1 - \gamma$. The multivariate polynomial

$$\mathbf{p}^*(x_1, \ldots, x_n) = p^*\Big(\frac{x_1 + \cdots + x_n}{n}\Big)$$

satisfies $|\mathbf{p}^*(\mathbf{x}) - \mathrm{OR}(\mathbf{x})| \le \gamma$ for all $\mathbf{x} \in \{-1,1\}^n$. Expanding $\mathbf{p}^*$ in the Fourier basis, we can write $\mathbf{p}^*(\mathbf{x}) = \sum_{|S| \le k} \hat{\mathbf{p}}^*_S \chi_S(\mathbf{x})$, where $\chi_S(\mathbf{x}) = \prod_{i \in S} x_i$. Then

$$\begin{aligned}
\mathbb{E}[\mathrm{OR}(X)] - \mathbb{E}[\mathrm{OR}(Y)] &\le \mathbb{E}[\mathbf{p}^*(X) + \gamma] - \mathbb{E}[\mathbf{p}^*(Y) - \gamma] \\
&\le 2\gamma + \mathbb{E}[\mathbf{p}^*(X)] - \mathbb{E}[\mathbf{p}^*(Y)] \\
&= 2\gamma + \sum\nolimits_{|S| \le k} \hat{\mathbf{p}}^*_S \cdot \big(\mathbb{E}[\chi_S(X)] - \mathbb{E}[\chi_S(Y)]\big) \\
&\le 2\gamma + \sum\nolimits_{S} |\hat{\mathbf{p}}^*_S| \cdot 2\delta \\
&= 2\gamma + 2\delta \cdot \hat{\ell}_1(\mathbf{p}^*).
\end{aligned}$$

The second to last step holds because χ_S is a k-local distinguisher with range $\{-1,1\}$, so its distinguishing advantage is at most 2δ. From Facts 8 and 9 it follows that

$$\hat{\ell}_1(\mathbf{p}^*) \le \frac{1}{2 + j(p,n)} \cdot \Big(1 + \frac{2}{n}\Big)^k \cdot \hat{\ell}_1(p) \le \frac{1}{2} \cdot \Big(\frac{n+2}{n}\Big)^k \cdot \hat{\ell}_1(p) \le \frac{e^2}{2} \cdot \hat{\ell}_1(p).$$

for $k < n$ as desired. ◀

Proof of Fact 8. If all the coefficients of a polynomial q are positive then $\hat{\ell}_1(q) = q(1)$. For $p(x) = \sum_{i=0}^{k} c_i x^i$ let $\tilde{p}(x) = \sum_{i=0}^{k} |c_i| x^i$. Then

$$\hat{\ell}_1(\tilde{p}(|a|x + |b|)) = \tilde{p}(|a| + |b|) \leq (|a| + |b|)^k \tilde{p}(1) = (|a| + |b|)^k \hat{\ell}_1(p).$$

The coefficients of $\tilde{p}(|a|x + |b|)$ dominate those of $p(ax + b)$, so we can conclude that $\hat{\ell}_1(p(ax + b)) \leq \hat{\ell}_1(\tilde{p}(|a|x + |b|)) \leq (|a| + |b|)^k \hat{\ell}_1(p)$. ◀

Proof of Fact 9. If $p(x) = \sum_{i=0}^{k} c_i x^i$ then

$$\hat{\ell}_1(\mathbf{p}) \leq \sum_{i=0}^{d} \frac{|c_i|}{n^i} \hat{\ell}_1\big((x_1 + ... + x_n)^i\big) \leq \sum_{i=0}^{d} \frac{|c_i|}{n^i} \cdot n^i = \hat{\ell}_1(p),$$

where the second to last step holds because all of the coefficients in $(x_1 + \cdots + x_n)^i$ are nonnegative, so the weight is $(1 + \cdots + 1)^i = n^i$. ◀

2.2 Construction of approximating polynomials

Our approximating polynomials will take the form of a Chebyshev polynomial evaluated at an appropriately chosen monomial. We note that Servedio, Tan, and Thaler in [14] also use polynomials of this form to give a degree-weight tradeoff of polynomials approximating the OR function.

For the proof of Theorem 1, we set $p(x) = T_r(x^d)$, where $d = \lfloor k^2/20n \rfloor \geq 1$, $r = \lfloor k/d \rfloor$, and T_r is the Chebyshev polynomial of degree r.

The Chebyshev polynomials satisfy (1) $|T_r(x)| \leq 1$ for all $x \in [-1, 1]$, (2) $T_r(1) = 1$, (3) $T_r'(1) = r^2$, (4) $T_r''(x) \geq 0$ for all $x \geq 1$, and (5) $\hat{\ell}_1(T_r) \leq 2^{2r}$. The first four properties are well-known; we provide a short proof of the fifth.

Proof of Property 5. We use an alternate definition of the Chebyshev polynomial: $T_r(x) = \frac{r}{2} \sum_{i=0}^{r/2} \frac{(-1)^i}{r-i} \binom{r-i}{i} 2^{r-2i} x^{r-2i}$. Thus, we have:

$$\hat{\ell}_1(T_r) = \frac{r}{2} \sum_{i=0}^{r/2} \frac{1}{r-i} \binom{r-i}{i} 2^{r-2i} \leq \sum_{i=0}^{r/2} \binom{r-i}{i} 2^{r-2i} \leq 2^r \sum_{i=0}^{r/2} \binom{r}{i} \leq 2^{2r}.$$ ◀

From properties (2), (3), and (4) it follows that

$$j(p, n) = p(1 + 2/n) - 1 \geq p'(1) \cdot \frac{2}{n} = \frac{2dr^2}{n} \geq \frac{k^2}{2dn} \geq 10.$$

The second to last inequality holds since for our choice of parameters $d \leq k$, so $r \geq 1$, and therefore $r \geq k/2d$.

By property (5), $\hat{\ell}_1(p) = 2^{2r} \leq 2^{2k/d}$. We now show this is at most $2^{80n/k}$. When $k^2 \geq 20n$, $d \geq k^2/40n$ and so $\hat{\ell}_1(p) \leq 2^{80n/k}$. Otherwise, $d = 1$, and $\hat{\ell}_1(p)$ is at most $2^{2k} \leq 2^{40n/k}$.

Finally, by property (1) $|p(x)| \leq 1$ for $|x| \leq 1$. By Lemma 7,

$$\mathbb{E}[\mathrm{OR}(X)] - \mathbb{E}[\mathrm{OR}(Y)] \leq \frac{2}{2 + 10} + e^2 \delta \cdot 2^{80n/k} \leq \frac{1}{3}$$

as long as $\delta \leq 2^{-80n/k}/6e^2$, proving Theorem 1.

3 Construction of distributions: Proof of Theorem 2

We reduce the existence of (k, ϵ)-indistinguishable distributions that can be reconstructed by the OR function to the analogous question for distributions of perfect bounded indistinguishability:

▶ **Lemma 10.** *For every $\varepsilon, N, K \leq N/2$ and $n \leq \epsilon N^2/121K^2$ the following holds. Assume there exist $(\sqrt{\epsilon n}, 0)$-wise indistinguishable distributions X, Y over $\{0,1\}^n$. Then there exist distributions X', Y' over $\{0,1\}^N$ such that $\mathbb{E}[\mathrm{OR}(X')] = \mathbb{E}[\mathrm{OR}(X)]$, $\mathbb{E}[\mathrm{OR}(Y')] = \mathbb{E}[\mathrm{OR}(Y)]$, and X', Y' are $(K, 2^{-\Omega(\epsilon N/K)})$-wise indistinguishable.*

Proof. To sample from X' (resp. Y'), first select a random set of n "active" indices among the N choices. Then, sample a string from X (resp. Y) and fill in the n indices with the sampled bits. Fill in the remaining $N - n$ places with 0s. This process does not change the chance that OR accepts a string, so the reconstruction error remains the same.

We now need to check that X', Y' are $(K, 2^{-\Omega(\epsilon N/K)})$-wise indistinguishable. Let S be any subset of $\{1, \ldots, N\}$ of size K and E be the event that at most k of the active indices fall in S. Conditioned on E, the projections of X' and Y' on S contain at most k bits from X and Y, respectively, and are therefore perfectly indistinguishable. Therefore the distinguishing advantage of any test $T\colon \{0,1\}^S \to \{0,1\}$ can be at most the probability that E does not occur.

To upper bound this probability, we take a union bound over all possible $\binom{K}{k}$ subsets of k active indices in S. For each such set, there is a probability of n/N that the first index is active, a probability of $(n-1)/(N-1)$ that the second index is active conditioned on the first one, and so on, obtaining:

$$\begin{aligned}\Pr[\text{there are at least } k \text{ active indices in } S] &\leq \binom{K}{k} \cdot \frac{n}{N} \cdot \frac{n-1}{N-1} \cdot \ldots \cdot \frac{n-k+1}{N-k+1} \\ &\leq \left(\frac{eK}{k} \cdot \frac{n}{N-k+1}\right)^k.\end{aligned}$$

Since $K \leq N/2$, we have that $k \leq N/2$ and $1/(N-k+1) \leq 2/N$. Plugging these estimates in, we conclude that the distinguishing advantage is at most $(2eKn/kN)^k$. We now set our parameters so that $\sqrt{\epsilon n} = k = 11nK/N$, implying that $(2eKn/kN)^k$ is upper bounded by $2^{-k} = 2^{-\Omega(nK/N)} = 2^{-\Omega(\epsilon N/K)}$. ◀

Now that we have reduced the problem of finding $(K, 2^{-\Omega(\epsilon N/K)})$-wise indistinguishable distributions to the one of finding $(\sqrt{\epsilon n}, 0)$-wise indistinguishable ones, for some specific n, we are ready to prove the following refinement of Theorem 2, which will be needed for the proof of Theorem 3.

▶ **Theorem 2'.** *For every ε, N, and $K \leq N/2$ there exists a pair of $(K, 2^{-\Omega(\epsilon N/K)})$-wise indistinguishable distributions X', Y' over $\{0,1\}^N$ such that $\mathbb{E}[\mathrm{OR}(X')] - \mathbb{E}[\mathrm{OR}(Y')] \geq 1 - \epsilon$.*

Proof. Corollary 2.2 in [3] shows the existence of $(\sqrt{\epsilon n}, 0)$-wise indistinguishable distributions X, Y over $\{0,1\}^n$ such that $\mathbb{E}[\mathrm{OR}(X)] - \mathbb{E}[\mathrm{OR}(Y)] = 1 - O(\epsilon)$. If $K \leq \sqrt{\epsilon N}/11$ the theorem follows directly from this Corollary. Otherwise, we apply Lemma 10 with $n = \lfloor \epsilon N^2/121K^2 \rfloor$ to X, Y and obtain the desired conclusion. ◀

4 Proof of Theorem 3

To prove Theorem 3 we reason as follows. Suppose there is a polynomial p of degree k such that $|p(x)| \leq 1$ for $|x| \leq 1$ and $p(1) = 1$. For $\varepsilon = p(1 + 2/n) - 1$, Theorem 2' and Lemma 7 together imply that

$$1 - \frac{\varepsilon}{6} \leq \mathbb{E}[\mathrm{OR}(X)] - \mathbb{E}[\mathrm{OR}(Y)] \leq \frac{2}{2+\varepsilon} + 2^{-\Omega(\varepsilon n/k)} \cdot \hat{\ell}_1(p),$$

from where we can conclude that $\hat{\ell}_1(p) \geq \Omega(\varepsilon) \cdot 2^{\Omega(\varepsilon n/k)}$, provided $\varepsilon \leq 1$. If the leading $\Omega(\varepsilon)$ term could be ignored, we would obtain Theorem 3 by taking the limit of the right-hand side as n goes to infinity and $\varepsilon n/2$ approaches $p'(1)$.

To account for the $\Omega(\varepsilon)$ term, we work with a carefully chosen, finite value of n. Our choice of n is sufficiently large so that the term $2^{\Omega(\varepsilon n/k)}$ dominates the term $\Omega(\varepsilon)$ in the expression lower bounding $\hat{\ell}_1(p)$, but sufficiently small so that $\varepsilon n/2$ is still lower bounded by $\Omega(p'(1))$. If n was a function of k only, this would be impossible as the value $\varepsilon = p(1+2/n)-1$ could even be negative. Our choice of n depends on the polynomial p itself via the parameters $\hat{\ell}_1(p)$ and $p'(1)$.

This description assumed that ε was at most one (or bounded by some fixed constant). The case of large ε can be handled along the same lines and is in fact technically easier.

Proof of Theorem 3. Let $p\colon \mathbb{R} \to \mathbb{R}$ be a degree-k polynomial such that $|p(x)| \leq 1$ for all $|x| \leq 1$ and $p(1) = 1$. Let $\varepsilon = p(1 + 2/n) - 1$ for $n = 4k^2\hat{\ell}_1(p)/p'(1)$. Expanding $p(1 + 2/n)$ around 1 we obtain

$$p(1 + 2/n) = p(1) + \frac{p'(1)}{n/2} + \sum_{i \geq 2} \frac{1}{(n/2)^i} \cdot \frac{p^{(i)}(1)}{i!}$$

where $p^{(i)}(1)$ is the i-th derivative of p at 1. Since $p(1) = 1$ it follows that

$$\varepsilon = p(1 + 2/n) - p(1) \geq \frac{p'(1)}{n/2} - \sum_{i \geq 2} \frac{1}{(n/2)^i} \cdot \frac{|p^{(i)}(1)|}{i!}.$$

A calculation of the derivatives shows that $|p^{(i)}|/i! \leq \binom{k}{i} \cdot \hat{\ell}_1(p)$ and so

$$\varepsilon \geq \frac{p'(1)}{n/2} - \sum_{i \geq 2} \frac{\binom{k}{i}}{(n/2)^i} \cdot \hat{\ell}_1(p) = \frac{p'(1)}{n/2} - \hat{\ell}_1(p) \cdot \sum_{i \geq 2} \left(\frac{k}{n/2}\right)^i \cdot \frac{1}{i!}. \tag{1}$$

Since $p'(1) \leq \hat{\ell}_1(p') \leq k\hat{\ell}_1(p)$, n is at least $4k$ and

$$\sum_{i \geq 2} \left(\frac{k}{n/2}\right)^i \cdot \frac{1}{i!} \leq \left(\frac{k}{n/2}\right)^2 \cdot \sum_{i \geq 2} \frac{1}{i!} \leq \left(\frac{k}{n/2}\right)^2.$$

From (1) we obtain that

$$\varepsilon \geq \frac{p'(1)}{n/2} - \hat{\ell}_1(p) \cdot \frac{k^2}{(n/2)^2} \geq \frac{p'(1)}{n}. \tag{2}$$

where the second inequality follows from our choice of n.

By Lemma 7 for every pair of (k, δ)-wise indistinguishable distributions X and Y over $\{0,1\}^n$,

$$\mathbb{E}[\mathrm{OR}(X)] - \mathbb{E}[\mathrm{OR}(Y)] \leq \frac{2}{2+\varepsilon} + e^2\delta\hat{\ell}_1(p).$$

If $\varepsilon \leq 1$, by Theorem 2' there exist $(k, 2^{-c\varepsilon n/k})$-wise indistinguishable distributions X and Y such that $\mathbb{E}[OR(X)] - \mathbb{E}[OR(Y)] \geq 1 - \varepsilon/6$. Setting $\delta = 2^{-c\varepsilon n/k}$ we obtain

$$e^2\delta\hat{\ell}_1(p) \geq 1 - \frac{\varepsilon}{6} - \frac{2}{2+\varepsilon} \geq \left(1 - \frac{\varepsilon}{6}\right) - \left(1 - \frac{\varepsilon}{3}\right) = \frac{\varepsilon}{6}.$$

Using inequality (2) and the definition of n we have

$$6e^2\hat{\ell}_1(p) \geq \varepsilon \cdot 2^{c\varepsilon n/k} \geq \frac{p'(1)}{n} \cdot 2^{cp'(1)/2k} = \frac{1}{\hat{\ell}_1(p)} \cdot \left(\frac{p'(1)}{2k}\right)^2 \cdot 2^{cp'(1)/2k}.$$

After rearranging terms we obtain

$$\hat{\ell}_1(p) \geq \frac{1}{\sqrt{6}e} \cdot \frac{p'(1)}{2k} \cdot 2^{cp'(1)/4k}.$$

Since $\hat{\ell}_1(p)$ is also at least $p(1) = 1$, it follows that $2^{\Omega(p'(1)/4k)}$, proving the theorem when $\varepsilon \leq 1$.

If $\varepsilon > 1$, by Theorem 2 there exist $(k, 2^{-cn/k})$-wise indistinguishable distributions X and Y such that $\mathbb{E}[\mathrm{OR}(X)] - \mathbb{E}[\mathrm{OR}(Y)] \geq 5/6$ and $e^2\delta\hat{\ell}_1(p) \geq 5/6 - 2/3 \geq 1/6$. Setting $\delta = 2^{-cn/k}$,

$$\hat{\ell}_1(p) \geq \frac{1}{6e^2} \cdot 2^{cn/k} \geq 2^{cp'(1)/2k}$$

using (2) and the assumption $\varepsilon > 1$. ◀

5 AND-OR formulas and perfect reconstruction

In this section we prove Theorem 4, give a variant with better parameters that provides almost-perfect reconstruction, and show that Theorem 4 is tight in the depth-2 case with respect to all uniform read-once AND ∘ OR formulas.

We first extend Lemma 10 to AND ∘ OR formulas of depth d:

▶ **Lemma 11.** *Assume there exist $(k, 0)$-wise indistinguishable distributions X and Y over $\{0,1\}^n$ for a regular depth-d AND ∘ OR tree f over n variables and with lowest-level fan-in m. Then there exist $(K, 2^{-\Omega(k)})$-wise indistinguishable distributions X' and Y' over $\{0,1\}^N$, $N = nM/m$, such that $\mathbb{E}[f'(X')] = \mathbb{E}[f(X)]$ and $\mathbb{E}[f'(Y')] = \mathbb{E}[f(Y)]$, where f' is a function taking the same form as f, except for that the lowest-level fan-in is M, provided $m \leq M/2$, $k/K \geq 4m/M$ and $n \leq m \cdot 2^{m-1}$.*

Proof. To sample from X' (resp. Y'), first sample a string from X (resp. Y), then extend each of the blocks with $M - m$ zeros positioned uniformly at random. Call the indices i in which X'_i inherits some bit of X *active*.

The distribution on active indices of X' can be described in the following alternative manner: First, choose each index i to be potentially active independently at random with probability $p = 2m/M$. If any block of X' has fewer than m potentially active indices, declare failure (F). Conditioned on not failing ($\overline{F}$), choose the active indices in each block uniformly at random among the potentially active ones.

Now let S be any set consisting of at most K inputs of f'. Let B be the event that S contains more than $2pK$ active indices. By Chernoff and union bounds,

$$\Pr[B|\overline{F}] \leq \frac{\Pr[B]}{1 - \Pr[F]} \leq \frac{2^{-pK}}{1 - (n/m)2^{-m}} \leq 2^{-k+1}$$

by our choice of parameters. As in the proof of Lemma 10 we conclude that X' and Y' must satisfy the conclusion. ◀

Proof of Theorem 4. We will use N and K to denote the quantities n and k from the statement of the theorem. By [15] and [3] there exist X and Y that are $(k, 0)$-wise indistinguishable but perfectly reconstructible by $\mathrm{AND}_{n^{1/2d-1}} \circ \mathrm{OR}_{n^{2/2d-1}} \circ \ldots \circ \mathrm{OR}_m$ for $m = n^{2/2d-1}$ and $k = \Omega(n^{\frac{d-1}{2d-1}})$. We will assume $K \geq N^{\frac{d-1}{2d-1}}$, for otherwise there is nothing left to prove. Set $k = (N/CK)^{1-1/d}$ for a sufficiently large constant C and $M = 4Km/k$. If $m > M/2$ then the conclusion follows directly from [15, 3] (as K will be at most a constant in terms of d times $N^{\frac{d-1}{2d-1}}$). If $n > m \cdot 2^{m-1}$, $(N/K)^{1-1/d}$ is upper bounded by a constant so the conclusion holds trivially. Otherwise, the statement of the theorem follows from Lemma 11. ◀

In the case $d = 2$, the parameters in Theorem 4 are the best possible, up to logarithmic terms, for all read-once CNFs. (The regime $k < n^{1/3}$ is resolved in [11, 3].)

▶ **Theorem 12.** *There exists a constant c such that for any n and $k \geq n^{1/3}$, no read-once CNF over n variables can perfectly reconstruct from any pair of $(k, 2^{-\Omega((n/k)^{1/2}\log^2 n)})$-wise indistinguishable distributions.*

The proof is omitted from this version owing to space limitations. If an exponentially small reconstruction error is acceptable, much better parameters for the underlying distributions are achievable:

▶ **Proposition 13.** *For all n, m, and k, the function $\mathrm{AND}_{n/m} \circ \mathrm{OR}_m$ can reconstruct from some pair of $(k, 2^{-\Omega(m/k)})$-wise indistinguishable distributions with error at most $2^{-\Omega(n/m)}$.*

Proof. Bun and Thaler [5] (improving work of Beigel) showed, via the connection in [3], that $\mathrm{AND}_{n/m} \circ \mathrm{OR}_m$ can reconstruct from some pair of $(\sqrt{m}, 0)$-wise indistinguishable distributions with error at most $2^{-\Omega(n/m)}$. We apply Lemma 11 with $k = \sqrt{m}$ and $M = mK/k$. ◀

6 A threshold weight lower bound

Proof of Corollary 6. Let X and Y be $(k, 2^{-\Omega((n/k)^{1-1/d})})$-wise indistinguishable distributions that the function f from Theorem 4 can perfectly reconstruct from. If p is a degree-k polynomial such that $|f(x) - p(x)| \leq 1/2 - 2^{-t}$ then by Lemma 7,

$$1 = \mathbb{E}[f(X)] - \mathbb{E}[f(Y)] \leq (1 - 2^{1-t}) + 2^{1-\Omega((n/k)^{1-1/d})} \cdot \hat{\ell}_1(p),$$

from where $\hat{\ell}_1(p) \geq 2^{\Omega((n/k)^{1-1/d})-t}$. Setting $t = c \cdot (n/k)^{1-1/d}$ gives the desired lower bound on $\hat{\ell}_1(p)$. ◀

The *threshold weight* of a function is its minimum degree-k threshold weight over all k. A result of Krause [9] (see also Lemma 27 in [5]) can be used to convert lower bounds on degree-k threshold weight into ones independent of degree.

▶ **Fact 14** (Krause, 2005). *For $f\colon \{0,1\}^n \to \{0,1\}$, let $F\colon \{0,1\}^{3n} \to \{0,1\}$ be given by $F(x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_n) = f(\ldots, (\overline{z_i} \text{ AND } x_i) \text{ OR } (z_i \text{ AND } y_i), \ldots)$. For any k, if f requires degree-k threshold weight w then F requires threshold weight $\sqrt{\min\{w/2n, 2^k\}}$.*

Applying Fact 14 to Corollary 6, we obtain a linear-size depth-d family of formulas that requires threshold weight $2^{\Omega(n^{1/2-1/(4d-6)})}$ on inputs of length n, matching Sherstov's bound for formulas [15].

Acknowledgments. We thank Mark Bun for telling us about the work of Servedio, Tan, and Thaler and for his advice on polynomial approximations, and the ICALP 2017 reviewers for several helpful suggestions.

References

1 Richard Beigel. The polynomial method in circuit complexity. In *8th Structure in Complexity Theory Conference*, pages 82–95. IEEE, 1993.

2 Richard Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4:339–349, 1994. doi:10.1007/BF01263422.

3 Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. In *CRYPTO*, 2016.

4 Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and markov-bernstein inequalities. In *Coll. on Automata, Languages and Programming (ICALP)*, pages 303–314, 2013.

5 Mark Bun and Justin Thaler. Hardness amplification and the approximate degree of constant-depth circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 2013.

6 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

7 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986. doi:10.1145/12130.12132.

8 Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014. doi:10.1137/120897432.

9 M. Krause. On the computational power of boolean decision lists. In *Computational Complexity*, pages 362–375, 2005.

10 Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.

11 Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

12 Moni Naor and Adi Shamir. Visual cryptography. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 1994.

13 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

14 R. Servedio, L-Y. Tan, and J. Thaler. Attribute-efficient learning and weight-degree tradeoffs for polynomial threshold functions. In *COLT*, 2012.

15 Alexander A. Sherstov. Breaking the Minsky-Papert barrier for constant-depth circuits. In *ACM Symp. on the Theory of Computing (STOC)*, pages 223–232, 2014.

16 Alexander A. Sherstov. The power of asymmetry in constant-depth circuits. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, 2015.

17 Robert Špalek. A dual polynomial for OR. *CoRR*, abs/0803.4516, 2008. URL: http://arxiv.org/abs/0803.4516.

18 Andrew Yao. Separating the polynomial-time hierarchy by oracles. In *26th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 1–10, 1985.

Finding Detours is Fixed-Parameter Tractable*†

Ivona Bezáková[1], Radu Curticapean[2], Holger Dell[3], and Fedor V. Fomin[4]

1 **Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA**
ib@cs.rit.edu

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
radu.curticapean@gmail.com

3 **Saarland University and Cluster of Excellence, MMCI, Saarbrücken, Germany**
hdell@mmci.uni-saarland.de

4 **University of Bergen, Bergen, Norway**
fomin@ii.uib.no

Abstract

We consider the following natural "above guarantee" parameterization of the classical Longest Path problem: For given vertices s and t of a graph G, and an integer k, the problem Longest Detour asks for an (s,t)-path in G that is at least k longer than a shortest (s,t)-path. Using insights into structural graph theory, we prove that Longest Detour is fixed-parameter tractable (FPT) on undirected graphs and actually even admits a single-exponential algorithm, that is, one of running time $\exp(O(k)) \cdot \mathrm{poly}(n)$. This matches (up to the base of the exponential) the best algorithms for finding a path of length at least k.

Furthermore, we study the related problem Exact Detour that asks whether a graph G contains an (s,t)-path that is exactly k longer than a shortest (s,t)-path. For this problem, we obtain a randomized algorithm with running time about $2.746^k \cdot \mathrm{poly}(n)$, and a deterministic algorithm with running time about $6.745^k \cdot \mathrm{poly}(n)$, showing that this problem is FPT as well. Our algorithms for Exact Detour apply to both undirected and directed graphs.

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

Keywords and phrases longest path, fixed-parameter tractable algorithms, above-guarantee parameterization, graph minors

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.54

1 Introduction

The Longest Path problem asks, given an undirected n-vertex graph G and an integer k, to decide whether G contains a path of length at least k, that is, a self-avoiding walk with at least k edges. This problem is a natural generalization of the classical NP-complete Hamiltonian Path problem, and the parameterized complexity community has paid exceptional attention to it. For instance, Monien [28] and Bodlaender [4] showed *avant*

* Full version at https://arxiv.org/abs/1607.07737.

† Most of this work was done while the authors were visiting the Simons Institute for the Theory of Computing. IB is supported by NSF grant CCF-1319987. RC is supported by ERC grant PARAMTIGHT (No. 280152).

la lettre that Longest Path is fixed-parameter tractable with parameter k and admits algorithms with running time $2^{O(k \log k)} n^{O(1)}$. This led Papadimitriou and Yannakakis [29] to conjecture that Longest Path is solvable in polynomial time for $k = \log n$, and indeed, this conjecture was resolved in a seminal paper of Alon, Yuster, and Zwick [2], who introduced the method of color coding and derived from it the first algorithm with running time $2^{O(k)} n$. Since this breakthrough of Alon et al. [2], the problem Longest Path occupied a central place in parameterized algorithmics, and several novel approaches were developed in order to reduce the base of the exponent in the running time [19, 22, 9, 8, 23, 33, 15, 15, 3]. We refer to two review articles in Communications of ACM [14, 24] as well as to the textbook [12, Chapter 10] for an extensive overview of parameterized algorithms for Longest Path. Let us however note that the fastest known randomized algorithm for Longest Path is due to Björklund et al. [3] and runs in time $1.657^k \cdot n^{O(1)}$, whereas the fastest known deterministic algorithm is due to Zehavi [34] and runs in time $2.597^k \cdot n^{O(1)}$.

In the present paper, we study the problem Longest Path from the perspective of an "above guarantee" parameterization that can attain small values even for long paths: For a pair of vertices $s, t \in V(G)$, we use $d_G(s,t)$ to denote the distance, that is, the length of a shortest path from s to t. We then ask for an (s,t)-path of length at least $d_G(s,t) + k$, and we parameterize by this offset k rather than the actual length of the path to obtain the problem Longest Detour. In other words, the first $d_G(s,t)$ steps on a path sought by Longest Detour are complimentary and will not be counted towards the parameter value. This reflects the fact that shortest paths can be found in polynomial time and could (somewhat embarrassingly) be much better solutions for Longest Path than the paths of logarithmic length found by algorithms that parameterize by the path length.

We study two variants of the detour problem, one asking for a detour of length at least k, and another asking for a detour of length exactly k.

Longest Detour	**Parameter:** k
Input: Graph G, vertices $s, t \in V(G)$, and integer k.	
Task: Decide whether there is an (s,t)-path in G of length at least $d_G(s,t) + k$.	

Exact Detour	**Parameter:** k
Input: Graph G, vertices $s, t \in V(G)$, and integer k.	
Task: Decide whether there is an (s,t)-path in G of length exactly $d_G(s,t) + k$.	

Our parameterization above the length of a shortest path is a new example in the general paradigm of "above guarantee" parameterizations, which was introduced by Mahajan and Raman [26]. Their approach was successfully applied to various problems, such as finding independent sets in planar graphs (where an independent set of size at least $\frac{n}{4}$ is guaranteed to exist by the Four Color Theorem), or the maximum cut problem, see e.g. [1, 11, 17, 16, 27].

Our results

We show the following tractability results for Longest Detour and Exact Detour:

- Longest Detour is fixed-parameter tractable (FPT) on undirected graphs. The running time of our algorithm is single-exponential, i.e., it is of the type $2^{O(k)} \cdot n^{O(1)}$ and thus asymptotically matches the running time of algorithms for Longest Path. Our approach requires a non-trivial argument in graph structure theory to obtain the single-exponential algorithm; a mere FPT-algorithm could be achieved with somewhat less effort. It should also be noted that a straightforward reduction rules out a running time of $2^{o(k)} \cdot n^{O(1)}$ unless the exponential-time hypothesis of Impagliazzo and Paturi [20] fails.

- Exact Detour is FPT on directed and undirected graphs. Actually, we give a polynomial-time Turing reduction from Exact Detour to the standard parameterization of Longest Path, in which we ask on input u, v and $k \in \mathbf{N}$ whether there is a (u, v)-path of length k. This reduction only makes queries to instances with parameter at most $2k + 1$. Pipelined with the fastest known algorithms for Longest Path mentioned above, this implies that Exact Detour admits a bounded-error randomized algorithm with running time $2.746^k n^{O(1)}$, and a deterministic algorithm with running time $6.745^k n^{O(1)}$.

By a self-reducibility argument, we also show how to construct the required paths rather than just detect their existence. This reduction incurs only polynomial overhead.

Techniques

The main idea behind the algorithm for Longest Detour is the following combinatorial theorem, which shows the existence of specific large planar minors in large-treewidth graphs while circumventing the full machinery used in the Excluded Grid Theorem [31]. Although the Excluded Grid Theorem already shows that graphs of sufficiently large treewidth contain arbitrary fixed planar graphs, resorting to more basic techniques allows us to show that linear treewidth suffices for our specific cases. More specifically, we show that there exists a global constant $c \in \mathbf{N}$ such that every graph of treewidth at least $c \cdot k$ contains as a subgraph a copy of a graph $K_4^{\geq k}$, which is any graph obtained from the complete graph K_4 by replacing every edge by a path with at least k edges. The proof of this result is based on the structural theorems of Leaf and Seymour [25] and Raymond and Thilikos [30].

With the combinatorial theorem at hand, we implement the following win/win approach: If the treewidth of the input graph is less than $c \cdot k$, we use known algorithms [5, 15] to solve the problem in single-exponential time. Otherwise the treewidth of the input graph is at least $c \cdot k$ and there must be a $K_4^{\geq k}$, which we use to argue that any path visiting the same two-connected component as $K_4^{\geq k}$ can be prolonged by rerouting it through $K_4^{\geq k}$. To this end, we set up a fixed system of linear inequalities corresponding to the possible paths in $K_4^{\geq k}$ such that rerouting is possible if and only if the system is unsatisfiable. We then verify the unsatisfiability of this fixed system by means of a computer-aided proof (more specifically, a linear programming solver). From LP duality, we also obtain a short certificate for the unsatisfiability, which we include in the full version of this extended abstract.

The algorithm for Exact Detour is based on the following idea. We run breadth-first search (BFS) from vertex v to vertex u. Then, for every (u, v)-path P of length $d_G(u, v) + k$, all but at most k levels of the BFS-tree contain exactly one vertex of P. Using this property, we are able to devise a dynamic programming algorithm for Exact Detour, provided it is given access to an oracle for Longest Path.

The remaining part of the paper is organized as follows. Section 2 contains definitions and preliminary results used in the technical part of the paper. In Section 3, we give an algorithm for Longest Detour while Section 4 is devoted to Exact Detour. Due to space constraints, we defer some proofs, some figures, a search-to-decision reduction for Longest Detour and Exact Detour to the full version of this extended abstract. Statements whose proofs are omitted here are marked with $\star$.

2 Preliminaries

We consider graphs G to be undirected, and we denote by uv an undirected edge joining vertices $u, v \in V(G)$. A *path* is a self-avoiding walk in G; the *length* of the path is its number of edges. An (s, t)-path for $s, t \in V(G)$ is a path that starts at s and ends at t. We allow

paths to have length 0, in which case $s = t$ holds. For a vertex set $X \subseteq V(G)$, denote by $G[X]$ the subgraph induced by X.

Tree decompositions. A *tree decomposition* $\mathcal{T}$ of a graph G is a pair $(T, \{X_t\}_{t \in V(T)})$, where T is a tree in which every node t is assigned a vertex subset $X_t \subseteq V(G)$, called a bag, such that the following three conditions hold:

(T1) Every vertex of G is in at least one bag, that is, $V(G) = \bigcup_{t \in V(T)} X_t$.

(T2) For every $uv \in E(G)$, there exists a node $t \in V(T)$ such that X_t contains both u and v.

(T3) For every $u \in V(G)$, the set T_u of all nodes of T whose corresponding bags contain u, induces a connected subtree of T.

The *width* of the tree decomposition $\mathcal{T}$ is the integer $\max_{t \in V(T)} |X_t| - 1$, that is, the size of its largest bag minus 1. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the smallest possible width that a tree decomposition of G can have.

We will need the following algorithmic results about treewidth.

▶ **Proposition 1** ([6]). *There is a $2^{O(k)} \cdot n$ time algorithm that, given a graph G and an integer k, either outputs a tree decomposition of width at most $5k + 4$, or correctly decides that $\mathrm{tw}(G) > k$.*

▶ **Proposition 2** ([5, 15]). *There is an algorithm with running time $2^{O(\mathrm{tw}(G))} \cdot n^{O(1)}$ that computes a longest path between two given vertices of a given graph.*

Let us note that the running time of Proposition 2 can be improved to $2^{O(\mathrm{tw}(G))} \cdot n$ by making use of the matroid-based approach from [15].

Our main theorem is based on graph minors, and we introduce some notation here.

▶ **Definition 3.** A *topological minor model* of H in G is a pair of functions (f, p) with $f : V(H) \to V(G)$ and $p : E(H) \to 2^{E(G)}$ such that

1. f is injective, and
2. for every edge $uv \in E(H)$, the graph $G[p(uv)]$ is a path from $f(u)$ to $f(v)$ in G, and
3. for edges $e, g \in E(H)$ with $e \neq g$, the paths $G[p(e)]$ and $G[p(g)]$ intersect only in endpoints or not at all.

The graph T *induced by* the topological minor model (f, p) is the subgraph of G that consists of the union of all paths $G[p(uv)]$ over all $uv \in E(H)$. The vertices in $f(V(H))$ are the *branch vertices* of T, and $G[p(e)]$ *realizes* the edge e in T.

3 Win/Win algorithm for Longest Detour

Throughout this section, let G be an undirected graph with n vertices and m edges, let $s, t \in V(G)$ and $k \in \mathbb{N}$. We wish to decide in time $2^{O(k)} \cdot n^{O(1)}$ whether G contains an (s, t)-path of length at least $d_G(s, t) + k$. To avoid trivialities, we assume without loss of generality that G is connected and $s \neq t$ holds. Moreover, we can safely remove vertices v that are not part of any (s, t)-path.

▶ **Definition 4.** Let G be a graph and let $s, t \in V(G)$. The *(s, t)-relevant part of G* is the graph induced by all vertices contained in some (s, t)-path. We denote it by $G_{s,t}$.

The graph $G_{s,t}$ can be computed efficiently from the block-cut tree of G. Recall that the *block-cut tree* of a connected graph G is a tree where each vertex corresponds to a *block*, that is, a maximal biconnected component $B \subseteq V(G)$, or to a *cut vertex*, that is, a vertex whose removal disconnects the graph. A block B and a cut vertex v are adjacent in the block-cut tree if and only if there is a block B' such that $B \cap B' = \{v\}$.

▶ **Lemma 5** (⋆). *Let B_s and B_t denote the blocks of G that contain s and t, respectively. Furthermore, let P be the unique (B_s, B_t)-path in the block-cut tree of G. Then $G_{s,t}$ is the graph induced by the union of all blocks visited by P.*

We formulate an immediate implication of Lemma 5 that will be useful later.

▶ **Corollary 6.** *The block-cut tree of $G_{s,t}$ is a (B_s, B_t)-path.*

Hopcroft and Tarjan [18] proved that the block-cut tree of a graph can be computed in linear time using DFS. Hence we obtain an algorithm for computing $G_{s,t}$ from G.

▶ **Corollary 7.** *There is a linear-time algorithm that computes $G_{s,t}$ from G.*

3.1 The algorithm

By definition, the graph $G_{s,t}$ contains the same set of (s,t)-paths as G. Our algorithm for Longest Detour establishes a "win/win" situation as follows: We prove that, if the treewidth of $G_{s,t}$ is "sufficiently large", then (G, s, t, k) is a YES-instance of Longest Detour. Otherwise the treewidth is small, and we use a known treewidth-based dynamic programming algorithm for computing the longest (s,t)-path. Hence the algorithm builds upon the following subroutines:

1. The algorithm from Corollary 7, computing the relevant part $G_{s,t}$ of G in time $O(n+m)$.
2. Compute Treewidth(G, w) from Proposition 1, which is given G and $w \in \mathbb{N}$ as input, and either constructs a tree-decomposition T of G whose width is bounded by $5w+4$, or outputs LARGE. If the algorithm outputs LARGE, then $\mathrm{tw}(G) > w$ holds. The running time is $2^{O(w)} \cdot n$.
3. Longest Path(G, T, s, t) from Proposition 2, which is given G, s, t and additionally a tree-decomposition T of G, and outputs a longest (s,t)-path in G. The running time is $2^{O(w)} \cdot n^{O(1)}$, where w denotes the width of T.

We now formalize what we mean by "sufficiently large" treewidth.

▶ **Definition 8.** A function $f : \mathbb{N} \to \mathbb{N}$ is *detour-enforcing* if, for all $k \in \mathbb{N}$ and all graphs G with vertices s and t, the following implication holds: If $\mathrm{tw}(G_{s,t}) > f(k)$, then G contains an (s,t)-path of length at least $d_G(s,t) + k$.

▶ **Theorem 9.** *The function $f : k \mapsto 32k + 2$ is detour-enforcing.*

We defer the proof of this theorem to the next section, and instead state Algorithm D, which uses f to solve Longest Detour. Algorithm D turns out to be an FPT-algorithm already when any detour-enforcing function f is known (as long as it is polynomial-time computable), and it becomes faster when detour-enforcing f of slower growth are used.

Algorithm D (Longest Detour) *Given (G, s, t, k), this algorithm decides whether the graph G contains an (s,t)-path of length at least $d_G(s,t) + k$.*

D1 (Restrict to relevant part) Compute $G_{s,t}$ using Corollary 7.
D2 (Compute shortest path) Compute the distance d between s and t in $G_{s,t}$.
D3 (Compute tree-decomposition) Call Compute Treewidth$(G_{s,t}, f(k))$.
  D3a (Small treewidth) If the subroutine found a tree-decomposition T of width at most $f(k)$, call Longest Path$(G_{s,t}, T, s, t)$. Output YES if there is an (s,t)-path of length at least $d + k$, otherwise output NO.
  D3b (Large treewidth) If the subroutine returned LARGE, output YES.

We prove the running time and correctness of Algorithm D.

▶ **Lemma 10 (⋆).** *For every polynomial-time computable detour-enforcing function $f : \mathbb{N} \to \mathbb{N}$, Algorithm D solves* Longest Detour *in time $2^{O(f(k))} \cdot n^{O(1)}$.*

Theorem 9 and Lemma 10 imply a $2^{O(k)} \cdot n^{O(1)}$ time algorithm for Longest Detour.

3.2 Overview of the proof of Theorem 9

In our proof of Theorem 9, large subdivisions of K_4 play an important role. Intuitively speaking, a sufficiently large subdivision of K_4 in $G_{s,t}$ allows us to route some (s,t)-path through it and then exhibit a long detour within that subdivision.

▶ **Definition 11.** For $k \in \mathbb{N}$, a graph F is a $K_4^{\geq k}$ if it can be obtained by subdividing each edge of K_4 at least k times. Please note that the numbers of subdivisions do not need to agree for different edges.

We show in Section 3.3 that graphs G containing $K_4^{\geq k}$ subgraphs in $G_{s,t}$ have k-detours.

▶ **Lemma 12.** *Let G be a graph and $k \in \mathbb{N}$. If $G_{s,t}$ contains a $K_4^{\geq k}$ subgraph, then G contains an (s,t)-path of length at least $d_G(s,t) + k$.*

Since the graph obtained by subdividing each edge of K_4 exactly k times is a planar graph on $O(k)$ vertices, the Excluded Grid Theorem yields a function $f : \mathbb{N} \to \mathbb{N}$ such that every graph of treewidth at least $f(k)$ contains some $K_4^{\geq k}$ minor. Furthermore, since every $K_4^{\geq k}$ has maximum degree 3, this actually shows that G contains some $K_4^{\geq k}$ as a *subgraph*. Thus, Lemma 12 implies that f is detour-enforcing, and a proof of this lemma immediately implies a weak version of Theorem 9.

By recent improvements on the Excluded Grid Theorem [7, 10], the function f above is at most a polynomial. However, even equipped with this deep result we cannot obtain a single-exponential algorithm for Longest Detour using the approach of Lemma 10: It would require f to be linear. In fact, excluding grids is too strong a requirement for us, since every function f obtained as a corollary of the full Excluded Grid Theorem must be super-linear [32]. We circumvent the use of the Excluded Grid Theorem and prove the following lemma from more basic principles.

▶ **Lemma 13.** *For graphs G and $k \in \mathbb{N}$, if $\mathrm{tw}(G) \geq 32k+2$, then G contains a $K_4^{\geq k}$ subgraph.*

Together, Lemmas 13 and 12 imply Theorem 9.

Proof of Theorem 9. Let G and $s,t \in V(G)$ and $k \in \mathbb{N}$ be such that $\mathrm{tw}(G_{s,t}) > f(k)$. By Lemma 13, the graph $G_{s,t}$ contains a $K_4^{\geq k}$ subgraph, so Lemma 12 implies that G contains an (s,t)-path of length $d_G(s,t) + k$. This shows that f is indeed detour-enforcing. ◀

3.3 Proof of Lemma 12: Rerouting in subdivided tetrahedra

Let (G,s,t,k) be an instance for Longest Detour such that $G_{s,t}$ contains a $K_4^{\geq k}$ subgraph M. We want to prove that $G_{s,t}$ has a path of length at least $d_G(s,t) + k$; in fact, we construct the desired detour entirely in the subgraph M, for which reason we first need to route some (s,t)-path through M.

▶ **Lemma 14 (⋆).** *There are two distinct vertices $u,v \in V(M)$ and two vertex-disjoint paths P_s and P_t in G such that P_s is an (s,u)-path, P_t is a (v,t)-path, and they only intersect with $V(M)$ at u and v.*

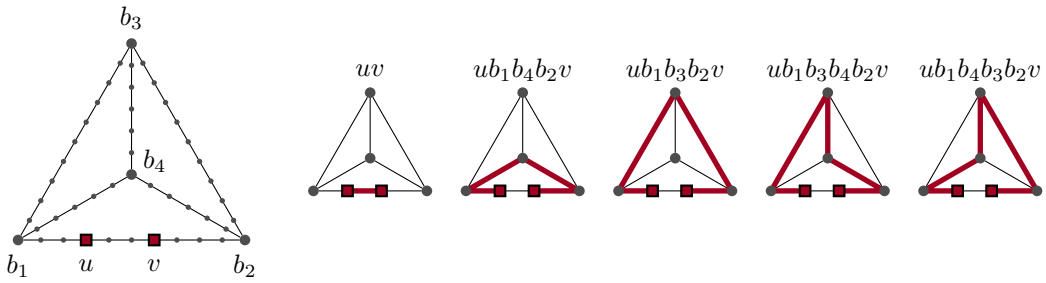


Figure 1 *Left:* One of the three possible cases for the relative positions of vertices u and v (*red squares*) in a subdivided tetrahedron $K_4^{\geq k}$ with degree-3 vertices $b_1, \ldots, b_4$ (*gray dots*) and at least $k = 5$ subdivision vertices (*small gray dots*). Here, u and v lie in the same subdivided edge. See the full version of the paper for the remaining cases. *Right:* An exhaustive list of all (u, v)-paths (*thick red*); Lemma 15 implies that the longest among them is at least k longer than the shortest one.

The proof of this lemma uses the fact that every block in the block-cut tree is biconnected. Next we show that every $K_4^{\geq k}$-graph M contains long detours.

▶ **Lemma 15.** *Let M be a $K_4^{\geq k}$-graph. For every two distinct vertices $u, v \in V(M)$, there is a (u, v)-path of length at least $d_M(u, v) + k$ in M.*

The proof idea is to distinguish cases depending on where u, v lie in M relative to each other. For each case, we can exhaustively list all (u, v)-paths (see Figure 1). We do not quite know the lengths of these paths, but we do know that each has length at least $d_M(u, v)$; moreover, each (b_i, b_j)-path in M for two distinct degree-3 vertices b_i and b_j has length at least k, since we subdivided K_4 at least k times. The claim of Lemma 15 is that one of the (u, v)-paths must have length at least $d_M(u, v)$. To prove this, we set up a linear program where the variables are $d_M(u, v)$, k, and the various path segment lengths; its infeasibility informs us that indeed a path that is longer by k must exist.

Proof Sketch for Lemma 15. Let M be a $K_4^{\geq k}$-graph, let $u, v \in V(M)$, and let $b_1, \ldots, b_4$ denote the four degree-3 vertices of M. Let P_u be a path in M that realizes an edge of K_4 and satisfies $u \in V(P_u)$, and let P_v be such a path with $v \in V(P_v)$. We distinguish three cases, one of which is depicted in Figure 1:

1. The two paths are the same, that is, $P_u = P_v$.
2. The two paths share a degree-3 vertex, that is, $|V(P_u) \cap V(P_v)| = 1$.
3. The two paths are disjoint, that is, $V(P_u) \cap V(P_v) = \emptyset$.

By the symmetries of K_4, this case distinction is exhaustive. Since K_4 has automorphisms that map any edge to any other edge, we can further assume that P_u is the path implementing the edge b_1b_2 such that P_u visits the vertices b_1, u, v, and b_2 in this order, see Figure 1.

We exhaustively list the set $\mathcal{P}$ of (u, v)-paths of M in Figure 1. Each path is uniquely specified by the sequence of the degree-3 vertices it visits. For example, consider the path $ub_1b_4b_2v$: This path consists of the four edge-disjoint segments ub_1, b_1b_4, b_4b_2, and b_2v; in the example figure, these segments have length 3, 6, 6, and 4, respectively. Given a path $P \in \mathcal{P}$, let $S(P)$ be the set of its segments between u, v, and the degree-3 vertices. For a path or a path segment s, we denote its length by $\ell(s)$.

Since M is a $K_4^{\geq k}$, every edge of K_4 is realized by a path of length at least k in M. Hence, $\ell(b_ib_j) \geq k$ holds for all i, j with $i \neq j$. Moreover, we have $\ell(b_1b_2) = \ell(b_1u) + \ell(uv) + \ell(vb_2)$ in case 1. Let $d = d_M(u, v)$; clearly $\ell(P) \geq d$ holds for all $P \in \mathcal{P}$. Our goal is to show that M has a (u, v)-path P with $\ell(P) \geq d + k$. To this end, we treat d, k, and all path

segment lengths $\ell(b_ib_j)$ for $i \neq j$ and $\ell(b_1u), \ell(uv), \ell(vb_2)$ as variables in a system of linear inequalities and establish that the claim holds if this system is unsatisfiable:

$$\ell(b_ib_j) \geq k\,, \qquad \text{for all } i,j \text{ with } i \neq j\,, \tag{1}$$

$$\ell(b_1u) + \ell(uv) + \ell(vb_2) = \ell(b_1b_2)\,, \tag{2}$$

$$\sum_{s \in S(P)} \ell(s) \geq d\,, \qquad \text{for all } P \in \mathcal{P}\,, \tag{3}$$

$$\sum_{s \in S(P)} \ell(s) \leq d + k - 1\,, \qquad \text{for all } P \in \mathcal{P}\,. \tag{4}$$

This system has eleven variables. Please note that d and k are also considered as variables in our formulation. The constraints in (1) express that M realizes each edge of K_4 by a path of length at least k. The constraints in (2) express that u and v lie on the path b_1b_2 and break it up into segments. The constraints in (3) express that no (u,v)-path is shorter than d in length, and the constraints in (4) express that every (u,v)-path has length strictly less than $d+k$. In the full version of this extended abstract, we prove that the linear program is infeasible, and so every setting for the variables that satisfies (1)–(3) must violate an inequality from (4); this means that M must contain a (u,v)-path of length at least $d+k$ in case 1. The proof is analogous when u and v are on different subdivided edges of the subdivided tetrahedron. We conclude that, no matter how u and v lie relative to each other in M, there is always a (u,v)-path that is at least k longer than a shortest one. ◀

This allows us to conclude Lemma 12 rather easily.

Proof of Lemma 12. Let $d = d_G(s,t)$ be the length of a shortest (s,t)-path in G. Let M be a $K_4^{\geq k}$ in $G_{s,t}$, and let P_s, P_t, u, and v be the objects guaranteed by Lemma 14. Let P_{uv} be a shortest (u,v)-path that only uses edges of M; its length is $d_M(u,v)$. Since the combined path P_s, P_{uv}, P_t is an (s,t)-path, its length is at least d.

Finally, Lemma 15 guarantees that there is a (u,v)-path Q_{uv} in M whose length is at least $d_M(u,v)+k$. Therefore, the length of the (s,t)-path P_s, Q_{uv}, P_t satisfies

$$\begin{aligned}\ell(P_s) + \ell(Q_{uv}) + \ell(P_t) &\geq \ell(P_s) + (d_M(u,v)+k) + \ell(P_t)\\ &= \ell(P_s) + \ell(P_{uv}) + \ell(P_t) + k \geq d + k\,.\end{aligned}$$

We constructed a path of at least length $d+k$ as required. ◀

3.4 Proof of Lemma 13: Large treewidth entails subdivided tetrahedra

To prove Lemma 13, we require some preliminaries from graph minors theory, among them a term for vertex sets that enjoy very favorable connectivity properties.

▶ **Definition 16** ([13])**.** Let G be a graph and $A, B \subseteq V(G)$. The pair (A,B) is a *separation* in G if the sets $A \setminus B$ and $B \setminus A$ are non-empty and no edge runs between them. The *order* of (A,B) is the cardinality of $A \cap B$.

For $S \subseteq V(G)$, we say that S is *linked* in G if, for every $X, Y \subseteq S$ with $|X| = |Y|$, there are $|X|$ vertex-disjoint paths between X and Y that intersect S exactly at its endpoints.

With these definitions at hand, we can adapt a result by Leaf and Seymour [25] to prove the following lemma on topological minor containment in graphs of sufficiently large treewidth. For any forest F on k vertices, with maximum degree 3, it asserts that graphs G of treewidth $\Omega(k)$ admit a separation such that one side contains F as a topological minor,

with the branch vertices of this topological minor being contained in $A \cap B$ and linked in G. We will use this lemma to complete the topological F-minor in $G[A]$ to a larger graph by using disjoint paths between vertices in $A \cap B$.

▶ **Lemma 17** ($\star$). *Let F be a forest on $k > 0$ vertices with maximum degree 3 and let G be a graph. If $\mathrm{tw}(G) \geq \frac{3}{2}k - 1$, then G has a separation (A, B) of order $|V(F)|$ such that:*

1. *There is a topological minor model (f, p) of F in $G[A]$.*
2. *For every vertex $v \in V(F)$ of degree ≤ 2, we have $f(v) \in A \cap B$.*
3. *$A \cap B$ is linked in $G[B]$.*

Building upon Lemma 17, we then prove Lemma 13 by adapting work of Raymond and Thilikos [30], who used a variant of Lemma 17 to prove the existence of k-wheel minors in graphs of treewidth $\Omega(k)$. To this end, let T and P be obtained by k-subdividing the full binary tree with 8 leaves, and the path with 8 vertices, respectively. We invoke Lemma 17 with F instantiated to the disjoint union $T \cup P$. Since F has $21k + 2$ vertices, we obtain from Lemma 17 that any graph G with $\mathrm{tw}(G) \geq 32k + 2 \geq \frac{3}{2} \cdot (21k + 2)$ has a separation (A, B) of order $|V(F)|$ that contains F in $G[A]$ and has $A \cap B$ linked in $G[B]$.

Let X_F denote the eight leaves of T, and let Y_F denote the eight non-subdivision vertices of P. Furthermore, let $X_G, Y_G \subseteq A \cap B$ denote the images of X_F and Y_F in $G[A]$ under a topological minor model guaranteed by Lemma 17. Since $A \cap B$ is linked, we can find eight disjoint paths connecting X_G and Y_G in $G[B]$. We then prove that, regardless of how these paths connect X_G and Y_G, they always complete the topological minor model of F to one of $K_4^{\geq k}$ in G. (The full version illustrates this in a figure.) Lemma 13 then follows.

Proof of Lemma 13. Let $k \in \mathbb{N}$ and let G be a graph with $\mathrm{tw}(G) \geq 32k + 2$. As before, let T denote the full binary tree with 8 leaves, with root r, after each edge was subdivided k times. Let P denote the path on 8 vertices after subdividing each edge k times.

We write $X_F = \{x_1, \ldots, x_8\}$ for the leaves of T, and we write $Y_F = \{y_1, \ldots, y_8\}$ for the vertices in P that were not obtained as subdivision vertices. Finally, we write F for the disjoint union $T \cup P$ and consider $X_F, Y_F \subseteq V(F)$. Note that $|V(F)| = 21k + 2$ and that the degree of all vertices in $X_F \cup Y_F$ is bounded by 2.

By Lemma 17, there is a separation (A, B) in G of order $|V(F)|$ such that $A \cap B$ is linked, and there is a topological minor model (f, p) of F in $G[A]$ with $f(X_F \cup Y_F) \subseteq A \cap B$. We write $X_G = \{f(v) \mid v \in X_F\}$ and $Y_G = \{f(v) \mid v \in Y_F\}$. In the following, we aim at completing the subgraph induced by (f, p) in G to a $K_4^{\geq k}$ subgraph.

Since $A \cap B$ is linked in $G[B]$, there are vertex-disjoint paths $L_1, \ldots, L_8$ between X_G and Y_G in $G[B]$ that avoid $A \cap B$ except at their endpoints. For $i \in [8]$, denote the endpoints of L_i in X_G and Y_G by s_i and t_i, respectively. Assume without limitation of generality (by reordering paths) that $t_i = f(y_i)$ holds for all $i \in [8]$. Furthermore, for $x \in X_G$, write $\sigma(x)$ for the vertex of Y_G that x is connected to via its path among $L_1, \ldots, L_8$.

Let S denote the image of T under (f, p), which is a tree; let $\mathit{root}(S) = f(r)$. Write S_1, S_2 for the two subtrees of S rooted at the children of $\mathit{root}(S)$. Let $\mathit{lca}(s_1, s_8)$ denote the lowest common ancestor of s_1 and s_8 in S. We distinguish two cases:

Case 1: We have $\mathit{lca}(s_1, s_8) \neq \mathit{root}(S)$. That is, s_1 and s_8 are both in S_1 or both in S_2. Assume without limitation of generality that $s_1, s_8 \in V(S_1)$, as the argument proceeds symmetrically otherwise. Let x and x' be two distinct leaves of S_2. Then we find a $K_4^{\geq k}$ in G by defining branch vertices $w = \mathit{lca}(s_1, s_8)$, $p = \mathit{lca}(x, x')$, $a = \sigma(x)$, and $b = \sigma(x')$. Note that $p \notin \{x, x'\}$ and that the four vertices are distinct.

We realize the edge pw along the (p, w)-path present in S, and ab along the (a, b)-path present in P. We realize pa by concatenating the (p, x)-path in S and the (x, a)-path in

$G[B]$, and we realize pb likewise. To realize wa, we proceed as follows: If a precedes b in the order on P, then concatenate the (w, s_1)-path in S with L_1 and the (y_1, a)-path in P. If b precedes a, then concatenate the (w, s_8)-path in S with L_8 and the (y_8, a)-path in P. Realize wb symmetrically. Then every edge between pairs in $\{w, p, a, b\}$ is realized, and it is so by a path of length at least k. This gives a topological minor model of $K_4^{\geq k}$ in G.

Case 2: We have $lca(s_1, s_8) = root(S)$. That is, s_1 and s_8 are in different subtrees S_1 and S_2. Let R be a subtree of height 2 in S that is disjoint from the (s_1, s_8)-path in S. It is easy to verify that such a subtree indeed exists; denote its root by p, its leaves by x, x', and its parent in S by w. Furthermore, define $a = \sigma(x)$ and $b = \sigma(x')$. We declare $\{w, p, a, b\}$ as branch vertices and connect them as in the previous case.

In both cases, the constructed topological minor model shows that G contains a $K_4^{\geq k}$ subgraph. This proves the lemma. ◀

4 Dynamic programming algorithm for exact detour

We devise an algorithm for Exact Detour using a reduction to Exact Path, the problem that is given (G, s, t, k) to determine whether there is an (s, t)-path of length exactly k.

▶ **Theorem 18 (⋆).** Exact Detour *is fixed-parameter tractable. In particular, it has a bounded-error randomized algorithm with running time* $2.746^k \operatorname{poly}(n)$*, and a deterministic algorithm with running time* $6.745^k \operatorname{poly}(n)$*.*

Before we state the algorithm, let us introduce some notation. Let $s, t \in V(G)$. For any $x \in V(G)$, we abbreviate $d_G(s, x)$, that is, the distance from s to x in G, with $d(x)$, and we let the i-th layer of G be the set of vertices x with $d(x) = i$. For $u, v \in V(G)$ with $d(u) < d(v)$, we write $G_{[u,v]}$ for the graph $G[X]$ induced by the vertex set X that contains u, v, and all vertices x with $d(u) < d(x) < d(v)$. We also write $G_{[u,\infty)}$ for the graph $G[X]$ induced by the vertex set X that contains u and all vertices x with $d(u) < d(x)$. These graphs can be computed in linear time using breadth-first search starting at s. We now describe an algorithm for Exact Detour that makes queries to an oracle for Exact Path.

The general idea is as follows. Let G be an undirected graph, and consider an (s, t)-path P of length $d + k$ where $d = d_G(s, t)$, and let x be a token that travels along this path from s to t. As the token advances one step in the path, the number $d(x)$ can be incremented, decremented, or stay the same. When x moves from s to t, we must increment $d(x)$ at least d times, can decrement it at most $k/2$ times, and keep it unchanged at most k times; the reason is that the path must reach t but must use exactly k edges more than a shortest path. The crucial observation is that there are at most k different layers whose intersection with the path P contains more than one vertex. The idea for the algorithm is to guess the layers with more than one vertex and run an algorithm for Exact Path on them.

Algorithm A (Exact Detour) *Given (G, s, t, k), this algorithm decides whether the graph G contains an (s, t)-path of length exactly $d_G(s, t) + k$.*

A1 (Initialize table) For each $x \in V(G)$ with $d(x) \leq d(t)$, set $T[x] = \emptyset$.
When the algorithm halts, every entry $T[x]$ of the table is meant to satisfy the following property Q_x: For each integer ℓ with $d(t) - d(x) \leq \ell \leq d(t) - d(x) + k$, the set $T[x]$ contains ℓ if and only if $G_{[x,\infty)}$ contains an (x, t)-path of length ℓ.

A2 (Compute entries for the last $k+1$ layers) For each $x \in V(G)$ with $d(t) - k \leq d(x) \leq d(t)$, let $T[x]$ be the set of all integers ℓ with $\ell \in \{0, \ldots, 2k\}$ such that there is an (x, t)-path of length ℓ in $G_{[x,\infty)}$ (that is, call Exact Path $(G_{[x,\infty)}, x, t, \ell)$).
When this step finishes, all vertices x in the last $k + 1$ layers satisfy property Q_x.

A3 (Inductively fill in earlier layers) For each d from $d(t)-k-1$ down to 0, for each x with $d(x)=d$, and for each y with $d(x)<d(y)\leq d(x)+k+1$, we do the following:

A3a Compute the set L of all $\ell' \in \{0,\ldots,2k+1\}$ such that there is an (x,y)-path of length ℓ' in $G_{[x,y]}$ (that is, call Exact Path $(G_{[x,y]},x,y,\ell')$).

A3b Set $T[x] := T[x] \cup (L + T[y])$.

We will show that, when all vertices of a layer d have been considered, all vertices x in the layers d and higher satisfy property Q_x.

A4 Accept if and only if $(d_G(s,t)+k) \in T[s]$ holds.

▶ **Lemma 19.** *Algorithm A is a polynomial-time Turing reduction from* Exact Detour *to* Exact Path*; on instances with parameter k, all queries have parameter at most $2k+1$.*

Proof. The running time of A is polynomially bounded since breadth-first search can be used to discover all partial graphs $G_{[x,y]}$ and $G_{[x,\infty)}$, and we loop at most over every pair of vertices in A2 and A3. For the parameter bound, note that the queries in A2 and A3 are for paths of length at most $2k$ and $2k+1$, respectively. It remains to prove the correctness.

We execute algorithm A on an instance (G,s,t,k). For the correctness, it suffices to prove that property Q_s holds at the end of the execution: Note that ℓ with $\ell = d_G(s,t)+k$ lies in the interval $[d(t)-d(s), d(t)-d(s)+k]$ since $d(s)=0$ and $d(t)=d_G(s,t)$ holds. Moreover, we have $G_{[s,\infty)}=G$. Thus Q_s guarantees that $\ell \in T[s]$ holds if and only if G contains an (s,t)-path of length ℓ, which by step A4 implies that A accepts if and only if (G,s,t,k) is a yes-instance of Exact Detour. Therefore it remains to prove that Q_s holds at the end of the execution of A. We do so using the following claim.

Claim: For all x with $0 \leq d(x) \leq d(t)$, property Q_x holds forever after the entry $T[x]$ is written to for the last time.

We prove this claim by induction on $d(x)$. For the base case, let x be a vertex with $d(x) \geq d(t)-k$. The entry $T[x]$ is only written to in step A2. To prove that Q_x holds after A2, let ℓ be an integer with $d(t)-d(x) \leq \ell \leq d(t)-d(x)+k$. Note that $d(t)-d(x) \geq 0$ and $d(t)-d(x)+k \leq d(t)-(d(t)-k)+k \leq 2k$ holds, and so step A2 adds ℓ to $T[x]$ if and only if the graph $G_{[x,\infty)}$ contains an (x,t)-path of length ℓ. Therefore, Q_x holds forever after A2 has been executed.

For the induction step, let x be a vertex with $d(x) < d(t)-k$. By the induction hypothesis, Q_y holds for all y with $d(y) > d(x)$. The entry $T[x]$ is only written to in step A3b, and when it is first written to, the outer d-loop in A3 has fully processed all layers larger than $d(x)$. Thus already when $T[x]$ is written to for the first time, Q_y holds for all y with $d(y) > d(x)$. Let T be the table right after A3b writes to $T[x]$ for the last time. It remains to prove that $T[x]$ satisfies Q_x. Let ℓ be an integer with $d(t)-d(x) \leq \ell \leq d(t)-d(x)+k$.

Claim: There is an (x,t)-path of length ℓ if and only if $T[x]$ contains ℓ.

For the forward direction, let P be an (x,t)-path in $G_{[x,\infty)}$ of length exactly ℓ. There are exactly ℓ vertices $u \in V(P) \setminus \{x\}$. Moreover, since every edge $uv \in E(P)$ satisfies $|d(u)-d(v)| \leq 1$, every $d \in \{d(x)+1,\ldots,d(t)\}$ must have some vertex $u \in V(P)$ with $d(u)=d$. Since $\ell \leq d(t)-d(x)+k$, there are at most k distinct d where more than one vertex $u \in V(P)$ satisfies $d(u)=d$. By the pigeon hole principle, there exists an integer d in the $(k+1)$-element set $\{d(x)+1,\ldots,d(x)+k+1\}$ such that there is exactly one vertex $y \in V(P)$ with $d(y)=d$.

Let $P_{[x,y]}$ be the subpath of P between x and y, and let ℓ' be its length. By construction, $P_{[x,y]}$ is an (x,y)-path in $G_{[x,y]}$. Moreover, we have $\ell' \leq \ell - (d(t)-d(y))$ since $V(P)\setminus\{x\}$ contains ℓ vertices u, at least $d(t)-d(y)$ of which satisfy $d(u) > d(y)$. By choice of ℓ and y, we obtain $\ell' \leq d(y)-d(x)+k \leq 2k+1$. For this setting of y and ℓ', step A3a detects the

path $P_{[x,y]}$ and ℓ' is added to the set L. The second piece $P_{[y,\infty]}$ of the path P is a (y,t)-path in $G_{[y,\infty]}$ of some length ℓ'' between $d(t)-d(y)$ and $d(t)-d(y)+k$; since Q_y holds when A3b is executed for x and y, the set $T[y]$ contains ℓ'', and so $\ell = \ell' + \ell''$ gets added to $T[x]$. Since elements never get removed from $T[x]$, the forward direction of the claim holds.

For the backward direction of the claim, assume that $T[x]$ contains ℓ. This means that ℓ is added in step A3b during the execution of the algorithm; in particular, consider the variables $y \in V(G)$, $\ell' \in L$, and $\ell'' \in T[y]$ when $\ell = \ell' + \ell''$ is added to $T[x]$. By the induction hypothesis, $\ell'' \in T[y]$ implies that there is a (y,t)-path in $G_{[y,\infty)}$ of length ℓ''. Moreover, ℓ' was set in A3a in such a way that there is an (x,y)-path of length ℓ' in the graph $G_{[x,y]}$. Combined, these two paths yield a single (x,t)-path in $G_{[x,\infty)}$ of length ℓ. The backward direction of the claim follows. ◀

The randomized algorithm of Björklund et al. [3] is for a variant of Exact Path where the terminal vertices s and t are not given, that is, any path of length exactly k yields a YES-instance. Their algorithm applies to our problem as well, with the same running time. We sketch an argument for this observation here. Recall that the idea is to reduce the problem to checking whether a certain polynomial is identically zero – this polynomial is defined by summing over all possible labelled walks of length k (see [12, Sec. 10.4.3]). We modify the polynomial by adding two leaf-edges, one incident to s and one to t, and restricting our attention only to $(k+2)$-walks that contain these two edges. The required information for such walks can still be computed efficiently as before. The crux of the proof is that walks that are not paths cancel out over a field of characteristic two; this argument works by a local re-orientation of segments of the walk – an operation that does not change the vertices of the walk and must therefore keep s and t fixed. The graph G contains a k-path if and only if the polynomial is not identically zero; this property remains true in our case. The rest of the argument goes through as before, so the algorithm of Björklund et al. applies to Exact Path with no significant loss in the running time.

The deterministic algorithm of Zehavi [34] also does not expect the terminal vertices to be given, but this algorithm works for the weighted version of the problem. In the weighted k-path problem, we are given a graph G, weights w_e on each edge, a number k, and a number W, and the question is whether there is a path of length exactly k such that the sum of all edge weights along the path is at most W. We observe the following simple reduction from Exact Path (with terminal vertices s and t) to the weighted k-path problem (without terminal vertices): Every edge gets assigned the same edge weight 2, except for the new leaf-edges at s and t, which get edge weight 1. Now every path with exactly $k+2$ edges has weight at most $W = 2k+2$ if and only if the first and the last edges of the path are the leaf-edges we added. Due to this reduction, Zehavi's algorithm applies to Exact Path with no significant loss in the running time.

Theorem 18 follows from Algorithm A by using either the algorithm of Björklund et al. [3] or Zehavi [34] as the oracle. We remark that Theorem 18 and Algorithm A apply to directed graphs as well, in which case an algorithm for Exact Path in directed graphs needs to be used (color coding yields the fastest randomized algorithm [2], while Zehavi's deterministic algorithm also applies to directed graphs).

Open problem. We conclude with an open problem: what is the complexity of Longest Detour in directed graphs? Neither directed treewidth nor cylindrical grid minors [21] seem to help. Can one even find an (s,t)-path of length $\geq d_G(s,t)+1$ in polynomial time?

Acknowledgments. We thank Daniel Lokshtanov, Meirav Zehavi, Petr Golovach, Saket Saurabh, Stephan Kreutzer, and Tobias Mömke for helpful discussions and answers.

References

1. Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. *Algorithmica*, 61(3):638–655, 2011. doi:10.1007/s00453-010-9428-7.
2. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
3. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *Journal of Computer and System Sciences*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.
4. Hans L. Bodlaender. On linear time minor tests with depth-first search. *Journal of Algorithms*, 14(1):1–23, 1993. doi:10.1006/jagm.1993.1001.
5. Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.
6. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016. doi:10.1137/130947374.
7. Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *Journal of the ACM*, 63(5):40:1–40:65, 2016. doi:10.1145/2820609.
8. Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM Journal on Computing*, 38(6):2526–2547, 2009. doi:10.1137/080716475.
9. Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 298–307. SIAM, 2007.
10. Julia Chuzhoy. Improved bounds for the excluded grid theorem. *CoRR*, abs/1602.02629, 2016. URL: http://arxiv.org/abs/1602.02629.
11. Robert Crowston, Mark Jones, Gabriele Muciaccia, Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Polynomial kernels for lambda-extendible properties parameterized above the Poljak-Turzik bound. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 43–54, 2013. doi:10.4230/LIPIcs.FSTTCS.2013.43.
12. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
13. Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.
14. Fedor V. Fomin and Petteri Kaski. Exact exponential algorithms. *Communications of the ACM*, 56(3):80–88, 2013. doi:10.1145/2428556.2428575.
15. Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 142–151, 2014. doi:10.1137/1.9781611973402.10.
16. Gregory Gutin, Eun Jung Kim, Michael Lampis, and Valia Mitsou. Vertex cover problem parameterized above and below tight bounds. *Theory of Computing Systems*, 48(2):402–410, 2011. doi:10.1007/s00224-010-9262-y.

17 Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. *Journal of Computer and System Sciences*, 78(1):151–163, 2012. doi:10.1016/j.jcss.2011.01.004.

18 John Hopcroft and Robert Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, June 1973. doi:10.1145/362248.362272.

19 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

20 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

21 Ken-ichi Kawarabayashi and Stephan Kreutzer. The directed grid theorem. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 655–664. ACM, 2015. doi:10.1145/2746539.2746586.

22 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-color. In *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 58–67, 2006. doi:10.1007/11917496_6.

23 Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125, pages 575–586. Springer, 2008. doi:10.1007/978-3-540-70575-8_47.

24 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Communications of the ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

25 Alexander Leaf and Paul D. Seymour. Tree-width and planar minors. *Journal of Combinatorial Theory, Series B*, 111:38–53, 2015. doi:10.1016/j.jctb.2014.09.003.

26 Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31(2):335–354, 1999. doi:10.1006/jagm.1998.0996.

27 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *Journal of Computer and System Sciences*, 75(2):137–153, 2009. doi:10.1016/j.jcss.2008.08.004.

28 Burkhard Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985. doi:10.1016/S0304-0208(08)73110-4.

29 Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. *Journal of Computer and System Sciences*, 53(2):161–170, 1996. doi:10.1006/jcss.1996.0058.

30 Jean-Florent Raymond and Dimitrios M. Thilikos. Low polynomial exclusion of planar graph patterns. *Journal of Graph Theory*, 84(1):26–44, 2017. doi:10.1002/jgt.22009.

31 Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.

32 Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994. doi:10.1006/jctb.1994.1073.

33 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Information Processing Letters*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

34 Meirav Zehavi. Mixing color coding-related techniques. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, volume 9294, pages 1037–1049. Springer, 2015. doi:10.1007/978-3-662-48350-3_86.

Further Approximations for Demand Matching: Matroid Constraints and Minor-Closed Graphs*

Sara Ahmadian[1] and Zachary Friggstad[†2]

1 **Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada**
sahmadian@uwaterloo.ca

2 **Department of Computing Science, University of Alberta, Edmonton, Canada**
zacharyf@ualberta.ca

Abstract

We pursue a study of the Generalized Demand Matching problem, a common generalization of the b-Matching and Knapsack problems. Here, we are given a graph with vertex capacities, edge profits, and asymmetric demands on the edges. The goal is to find a maximum-profit subset of edges so the demands of chosen edges do not violate the vertex capacities. This problem is **APX**-hard and constant-factor approximations are already known.

Our main results fall into two categories. First, using iterated relaxation and various filtering strategies, we show with an efficient rounding algorithm that if an additional matroid structure $\mathcal{M}$ is given and we further only allow sets $F \subseteq E$ that are independent in $\mathcal{M}$, the natural LP relaxation has an integrality gap of at most $\frac{25}{3} \approx 8.333$. This can be further improved in various special cases, for example we improve over the 15-approximation for the previously-studied Coupled Placement problem [Korupolu et al. 2014] by giving a 7-approximation.

Using similar techniques, we show the problem of computing a minimum-cost base in $\mathcal{M}$ satisfying vertex capacities admits a $(1, 3)$-bicriteria approximation: the cost is at most the optimum and the capacities are violated by a factor of at most 3. This improves over the previous $(1, 4)$-approximation in the special case that $\mathcal{M}$ is the graphic matroid over the given graph [Fukanaga and Nagamochi, 2009].

Second, we show Demand Matching admits a polynomial-time approximation scheme in graphs that exclude a fixed minor. If all demands are polynomially-bounded integers, this is somewhat easy using dynamic programming in bounded-treewidth graphs. Our main technical contribution is a sparsification lemma that allows us to scale the demands of some items to be used in a more intricate dynamic programming algorithm, followed by some randomized rounding to filter our scaled-demand solution to one whose original demands satisfy all constraints.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures, G.2.1 Combinatorial Algorithms, G.1.6 Optimization

Keywords and phrases Approximation Algorithms, Column-Restricted Packing, Demand Matching, Matroids, Planar Graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.55

* A full version of the paper is available at https://arxiv.org/abs/1705.10396.

† This research was undertaken, in part, thanks to funding from the Canada Research Chairs program and an NSERC Discovery Grant.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 55; pp. 55:1–55:13

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

1 Introduction

Many difficult combinatorial optimization problems involve resource allocation. Typically, we have a collection of resources, each with finite supply or **capacity**. Additionally there are tasks to be accomplished, each with certain requirements or **demands** for various resources. Frequently the goal is to select a maximum value set of tasks and allocate the required amount of resources to each task while ensuring we have enough resources to accomplish the chosen tasks. This is a very well-studied paradigm: classic problems include Knapsack, Maximum Matching, and Maximum Independent Set, and more recently-studied problems include Unsplittable Flow [1] and Coupled Placement [9]. In general, we cannot hope to get non-trivial approximation algorithms for these problems. Even the simple setting of Maximum Independent Set is inapproximable [8, 18], so research frequently focuses on well-structured special cases.

Our primary focus is when each task requires at most two different resources. Formally, in Generalized Demand Matching (GDM) we are given a graph $G = (V, E)$ with, perhaps, parallel edges. The vertices should be thought of as resources and the tasks as edges. Each $v \in V$ has a capacity $b_v \geq 0$ and each $uv \in E$ has demands $d_{u,e}, d_{v,e} \geq 0$ and a value $p_{uv} \geq 0$. A subset $M \subseteq E$ is *feasible* if $d_v(\delta(v) \cap M) \leq b_v$ for each $v \in V$ (we use $d_v(S)$ as shorthand for $\sum_{e \in S} d_{v,e}$ when $S \subseteq \delta(v)$). We note that the simpler term Demand Matching (DM) is used when $d_{u,e} = d_{v,e}$ for each edge $e = uv$ (e.g. [15, 17]).

DM is well-studied from the perspective of approximation algorithms. It is fairly easy to get constant-factor approximations and some work has been done refining these constants. Moreover the integrality gap of a natural LP relaxation is also known to be no worse than a constant (see the related work section). On the other hand, DM is **APX**-hard [15].

Our main results come in two flavours. First, we look to a generalization we call Matroidal Demand Matching ($\mathsf{GDM_M}$). Here, we are given the same input as in GDM but there is also a matroid $\mathcal{M} = (E, \mathcal{I})$ over the edges E with independence system $\mathcal{I} \subseteq 2^E$ that further restricts feasibility of a solution. A set $F \subseteq E$ is feasible if it is feasible as a solution to the underlying GDM problem and also $F \in \mathcal{I}$. We assume $\mathcal{M}$ is given by an efficient independence oracle. Our algorithms will run in time that is polynomial in the size of G and the maximum running time of the independence oracle.

As a special case, $\mathsf{GDM_M}$ includes the previously-studied Coupled Placement problem. In Coupled Placement, we are given a bipartite graph $G = (V, E)$ with vertex capacities. The tasks are not individual edges, rather for each task j and each $e = uv \in E$ we have demands $d^j_{u,e}, d^j_{v,e}$ placed on the respective endpoints u, v for placing j on edge e. Finally, each task j has a profit p_j and the goal is to select a maximum-profit subset of tasks j and, for each chosen task j, assign j to an edge of G so vertex capacities are not violated. We note that an edge may receive many different tasks. This can be viewed as an instance of $\mathsf{GDM_M}$ by creating parallel copies of each edge $e \in E$, one for each task j with corresponding demand values and profit for j and letting $\mathcal{M}$ be the partition matroid ensuring we take at most one edge corresponding to any task.

For another interesting case, consider an instance where, in addition to tasks requiring resources from a shared pool, each also needs to be connected to a nearby power outlet. We can model such an instance by letting $\mathcal{M}$ be a transversal matroid over a bipartite graph where tasks form one side, outlets form the other side, and an edge indicates the edge can reach the outlet.

In fact $\mathsf{GDM_M}$ can be viewed as a packing problem with a particular submodular objective function. These are studied in [2] so the problem is not new; our results are improved

approximations. Our techniques also apply to give bicriteria approximations for the variant of GDM_M where we must pack a cheap base of the matroid while obeying congestion bounds. In the special case where $\mathcal{M}$ is the graphic matroid over G itself (i.e. the Minimum Bounded-Congestion Spanning Tree problem), we get an improved bicriteria approximation.

Second, we study GDM in special graph classes. In particular, we demonstrate a PTAS in families of graphs that exclude a fixed minor. This is complemented by showing that even DM is strongly NP-hard in simple planar graphs, thereby ruling out a fully-polynomial time approximation scheme (FPTAS) in simple planar graphs unless $\mathbf{P} = \mathbf{NP}$.

1.1 Statements of Results and Techniques

We first establish some notation. For a matroid $\mathcal{M} = (E, \mathcal{I})$, we let $r_\mathcal{M} : 2^E \to \mathbb{Z}_{\geq 0}$ be the rank function for $\mathcal{M}$. We omit the subscript $\mathcal{M}$ if the matroid is clear from the context. For $v \in V$ we let $\delta(v)$ be all edges having v as one endpoint; for $F \subseteq E$ we let $\delta_F(v)$ denote $\delta(v) \cap F$. For a vector of values x indexed by a set S, we let $x(A) = \sum_{i \in A} x_i$ for any $A \subseteq S$. A **polynomial-time approximation scheme** (PTAS) is an approximation algorithm that accepts an additional parameter $\epsilon > 0$. It finds a $(1+\epsilon)$-approximation in time $O(n^{f(\epsilon)})$ for some function f (where n is the size of the input apart from ϵ), so the running time is polynomial for any constant $\epsilon > 0$. An FPTAS is a PTAS with running time being polynomial in $\frac{1}{\epsilon}$ and n.

We say an instance of GDM has a **consistent ordering of edges** if E can be ordered such that the restriction of this ordering to each set $\delta(v)$ has these edges $e \in E$ appear in nondecreasing order of demands $d_{v,e}$. For example, DM itself has a consistent ordering of demands, just sort edges by their demand values. This more general case was studied in [13]. We say the instance is **conflict-free** if for any $e, f \in E$ we have that $\{e, f\}$ does not violate the capacity of any vertex.

In the first half of our paper, we mostly study the following linear-programming relaxation of GDM_M. Here, $r : 2^E \to \mathbb{Z}$ is the rank function for $\mathcal{M}$.

$$\max : \left\{ \sum_{e \in E} p_e x_e : \sum_{e \in \delta(v)} d_{v,e} x_e \leq b_v \ \forall v \in V, \quad x(A) \leq r(A) \ \ \forall A \subseteq E, \quad x \geq 0 \right\} \quad \textbf{(LP-M)}$$

Note $x(\{e\}) \leq 1$ is enforced for each $e \in E$ as $r(\{e\}) \leq 1$. It is well-known that the constraints can be separated in polynomial time when given an efficient independence oracle for $\mathcal{M}$, so we can find an extreme point optimum solution to (**LP-M**) in polynomial time.

Throughout, we assume each edge is feasible by itself. This is without loss of generality: an edge that is infeasible by itself can be discarded[1]. We first prove the following.

▶ **Theorem 1.** *Let* $\mathrm{OPT}(\textbf{LP-M})$ *denote the optimum solution value of* (**LP-M**). *If* $d_{v,e} \leq b_v$ *for each* $v \in V, e \in \delta(v)$ *then we can find, in polynomial time, a feasible solution* $\mathsf{M} \subseteq E$ *such that* $\mathrm{OPT}(\textbf{LP-M})/p(\mathsf{M})$ *(and, thus, the integrality gap) is at most:*

- $\frac{25}{3}$ *in general graphs*
- 7 *in bipartite graphs*
- 5 *if the instance has a consistent ordering of edges*
- 4 *if the instance is conflict-free*
- $1 + O(\epsilon^{1/3})$ *if* $d_{v,e} \leq \epsilon \cdot b_v$ *for each* $v \in V, e \in \delta(v)$ *(i.e. edges are* ϵ***-small)***

[1] This is a standard step when studying packing LPs, even the natural Knapsack LP relaxation has an unbounded integrality gap if we consider items that do not fit by themselves.

These bounds also apply to graphs with parallel edges, so we get a 7-approximation for Coupled Placement, which beats the previously-stated 15-approximation in [9].

We prove all bounds in Theorem 1 using the same framework: iterated relaxation to find some $\mathsf{M}' \in \mathcal{I}$ with $p(\mathsf{M}') \geq OPT_{LP}$ that may violate some capacities by a controlled amount, followed by various strategies to pare the solution down to a feasible solution. We note constant-factor approximations for GDM_M were already implicit in [2], the bounds in Theorem 1 improve over their bounds and are relative to (**LP-M**) whereas [2] involves multilinear extensions of submodular functions.

Our techniques can also be used to address a variant of GDM_M. The input is the same, except we are required to select a base of $\mathcal{M}$. The goal is to find a minimum-value base satisfying the vertex capacities. More formally, let Minimum Bounded-Congestion Matroid Basis be given the same way as in GDM_M, except the goal is to find a minimum-cost base B of $\mathcal{M}$ satisfying the vertex capacities (i.e. the cheapest base that is a solution to the GDM_M problem).

When all demands are 1, this is the Minimum Bounded-Degree Matroid Basis problem which, itself, contains the famous Minimum Bounded-Degree Spanning Tree problem. As an important special case, we let Minimum Bounded-Congestion Spanning Tree denote the problem when $k = 2$ with arbitrary demands where $\mathcal{M}$ is the graphic matroid over G. Even determining if there is a feasible solution is **NP**-hard, so we settle with approximations that may violate the capacities a bit. Consider the following LP relaxation, which we write when G can even be a hypergraph.

$$\min : \left\{ \sum_{e \in E} p_e x_e : \sum_{e \in \delta(v)} d_{v,e} x_e \leq b_v \ \forall v \in V, \quad x(A) \leq r(A) \ \forall A \subseteq E, \quad x(E) = r(E), \quad x \geq 0 \right\} \tag{\textbf{LP-B}}$$

As a side effect of how we prove Theorem 1, we also prove the following.

▶ **Corollary 2.** *If G is a hypergraph where each edge has size at most k, then in polynomial time we can either determine there is no integral point in* (**LP-B**) *or we can find a base* B *of* $\mathcal{M}$ *such that* $p(\mathsf{B}) \leq \mathrm{OPT}(\textbf{LP-B})$ *and* $d_v(\delta_\mathsf{B}(v)) \leq b_u + k \cdot \max_{e \in \delta(v)} d_{v,e}$ *for each* $v \in V$.

▶ **Theorem 3.** *There is a* $(1, 1 + k)$*-bicriteria approximation for* Minimum Bounded-Congestion Matroid Basis.

In particular, there is a $(1, 3)$-bicriteria approximation for Minimum Bounded-Congestion Spanning Tree, beating the previous best $(1, 4)$-bicriteria approximation [5]. Theorem 3 matches the bound in [9] for the special case of Coupled Placement in k-partite hypergraphs, but in a more general setting.

One could also ask if we can generalize Theorem 1 to hypergraphs. An $O(k)$-approximation is already known [2] and the integrality gap of (**LP-M**) is $\Omega(k)$ even without matroid constraints, so we could not hope for an asymptotically better approximation. We remind the reader that our focus in GDM_M is improved constants in the case of graphs ($k = 2$).

Our second class of results are quite easy to state. We study GDM in families of graphs that exclude a fixed minor. It is easy to see GDM is strongly **NP**-hard in planar graphs if one allows parallel edges as it is even strongly **NP**-hard with just two vertices, *e.g.* see [6, 11]. We show the presence of parallel edges is not the only obstacle to getting an FPTAS for GDM (or even DM) in planar graphs.

▶ **Theorem 4.** DM *is NP-hard in simple, bipartite planar graphs even if all demands, capacities, values, and vertex degrees are integers bounded by a constant.*

We then present our main result in this vein, which gives a PTAS for GDM in planar graphs among other graph classes.

▶ **Theorem 5.** GDM *admits a PTAS in families of graphs that exclude a fixed minor.*

This is obtained through the usual reduction to bounded-treewidth graphs [4]. We would like to scale demands to be polynomially-bounded integers, as then it is easy to solve the problem using dynamic programming over the tree decomposition. But packing problems are too fragile for scaling demands naively: an infeasible solution may be regarded as feasible in the scaled instance.

We circumvent this issue with a sparsification lemma showing there is a near-optimal solution M' where, for each vertex v, after packing a constant number of edges across v the remaining edges in $\delta_{\mathsf{M}'}(v)$ have very small demand compared with even the residual capacity. Our dynamic programming algorithm then guesses these large edges in each bag of the tree decomposition and packs the remaining edges according to scaled values. The resulting solution may be slightly infeasible, but the blame rests on our scaling of *small* edges and certain pruning techniques can be used to whittle this solution down to a feasible solution with little loss in the profit.

1.2 Related Work

DM (the case with symmetric demands) is well-studied. Shepherd and Vetta initially give a 3.264-approximation in general graphs and a 2.764-approximation in bipartite graphs [15]. These are all with respect to the natural LP relaxation, namely (**LP-M**) with matroid constraints replaced by $x_e \leq 1, \forall e \in E$. They also prove that DM is **APX**-hard even in bipartite graphs and give an FPTAS in the case G is a tree.

Parekh [13] improved the integrality gap bound for general graphs to 3 in cases of GDM that have a consistent ordering of edges. Singh and Wu improve the gap in bipartite graphs to 2.709 [17]. The lower bound on the integrality gap for general graphs is 3 [15], so the bound in [13] is tight. In bipartite graphs, the gap is at least 2.699 [17].

Bansal, Korula, Nagarajan, and Srinivasan study the generalization of GDM to hypergraphs [2]. They show if each edge has at most k endpoints, the integrality gap of the natural LP relaxation is $\Theta(k)$. They also prove that a slight strengthening of this LP has a gap of at most $(e + o(1)) \cdot k$. Even more relevant to our results is that they prove if the value function over the edges is submodular, then rounding a relaxation based on the multilinear extension of submodular functions yields a $\left(\frac{e^2}{e-1} + o(1)\right) \cdot k$-approximation. For $k = 2$, this immediately gives a constant-factor approximation for GDM_M by considering the submodular objective function $f : 2^E \to \mathbb{R}$ given by $f(S) = \max\{p(S') : S' \subseteq S, S' \in \mathcal{I}\}$.

They briefly comment on the case $k = 2$ in their work and say that even optimizations to their analysis for this special case yields only a 11.6-approximation for DM (i.e. without a matroid constraint). So our $\frac{25}{3}$-approximation for GDM_M is an improvement over their work. They also study the case where $d_{v,e} \leq \epsilon \cdot b_v$ for each $v \in V$ and each hyperedge $e \in \delta(v)$ and present an algorithm for GDM with submodular objective functions whose approximation guarantee tends to $\frac{4e^2}{e-1}$ as $\epsilon \to 0$ (with k fixed).

As noted earlier, our results yield improvements for two specific problems. First, our 7-approximation for GDM_M in bipartite graphs improves over the 15-approximation for Coupled Placement [9]. The generalization of Coupled Placement to k-partite hypergraphs is also studied in [9] where they obtain an $O(k^3)$-approximation, but this was already inferior to the $O(k)$-approximation in [2] when viewing it as a submodular optimization problem with packing constraints.

Second, our work also applies to the Minimum-Congestion Spanning Tree problem, defined earlier. Determining if there is even a feasible solution is **NP**-hard as this models the Hamiltonian Path problem. A famous result of Singh and Lau shows if all demands are 1 (so we want to bound the degrees of the vertices) then we can find a spanning tree with cost at most the optimum cost (if there is any solution) that violates the degree bounds additively by $+1$ [16]. In the case of arbitrary demands, the best approximation so far is a $(1, 4)$-approximation [5]: it finds a spanning tree whose cost is at most the optimal cost and violates the capacities by a factor of at most 4. It is known that obtaining a $(1, c)$-approximation is **NP**-hard for any $c < 2$ [7].

2 Approximation Algorithm for Demand Matching Problem

Here we present approximation algorithms for GDM_M and prove Theorem 1 and Corollary 2. Our algorithm consists of two phases: the iterative relaxation phase and the pruning phase. The first finds a set $\mathsf{M}' \in \mathcal{I}$ with $p(\mathsf{M}') \geq \mathrm{OPT}(\textbf{LP-M})$ that places demand at most $b_v + 2 \cdot \max_{e \in \delta_{\mathsf{M}'}(v)} d_{v,e}$ on each $v \in V$. The second prunes M' to a feasible solution, different pruning strategies are employed to prove the various bounds in Theorem 1.

2.1 Iterative Relaxation Phase

This part is presented for the more general case of hypergraphs where each edge has at most k endpoints. Our GDM_M results in Theorem 1 pertain to $k = 2$, but we will use properties of this phase in our proof of Corollary 2. The algorithm starts with (**LP-M**) and iteratively removes edge variables and vertex capacities.

We use the following notation. For some $W \subseteq V, F \subseteq E$, a matroid $\mathcal{M}'$ with ground set F, and values $b'_v, v \in W$ we let $\textbf{LP-M}[W, F, \mathcal{M}', b']$ denote the LP relaxation we get from (**LP-M**) over the graph (V, F) with matroid $\mathcal{M}'$ where we drop capacity constraints for $v \in V - W$ and use capacities b'_v for $v \in W$.

Note that the relevant graph for $\textbf{LP-M}[W, F, \mathcal{M}', b']$ still has all vertices V, it is just that some of the capacity constraints are dropped. Also, for a matroid $\mathcal{M}'$ and an edge $e \in F$ we let $\mathcal{M}' - e$ be the matroid obtained by deleting e and, if $\{e\}$ is independent in $\mathcal{M}'$, we let $\mathcal{M}'/e$ be the matroid obtained by contracting e (i.e. a set A is independent in $\mathcal{M}'/e$ if and only if $A \cup \{e\}$ is independent in $\mathcal{M}'$).

Algorithm 1 describes the steps in the iterated relaxation phase. Correct execution and termination are consequences of the following two lemmas. Their proofs are standard for iterated techniques and are deferred to the full version.

▶ **Lemma 6.** *Throughout the execution of the algorithm, whenever $\mathcal{M}'$ is contracted by e we have $\{e\}$ is independent (i.e. e is not a loop) in $\mathcal{M}'$.*

▶ **Lemma 7.** *The algorithm terminates in polynomial time and the returned set M' is an independent set in $\mathcal{M}$ with $p(\mathsf{M}') \geq \mathrm{OPT}(\textbf{LP-M})$. Furthermore, if at any point $W' = \emptyset$ then the corresponding extreme point solution x^* is integral.*

The last statement in the lemma emphasizes the last case in the body of the loop cannot be encountered if $W' = \emptyset$.

Next, we prove M' is a feasible demand matching with respect to capacities $b_v + k \cdot \max_{e \in \delta(v)} d_{e,v}$ for each $v \in V$ by utilizing the following claim.

▶ **Claim 8.** *In any iteration, if $0 < x^*_e < 1$ for each $e \in F$ then $|\delta_F(v)| \leq x^*(\delta_F(v)) + k$ for some $v \in W$.*

Algorithm 1 Iterated Relaxation Procedure for $\mathsf{GDM}_{\mathsf{M}}$.

$W \leftarrow V, F \leftarrow E, \mathcal{M}' \leftarrow \mathcal{M}$
$b'_v \leftarrow b_v$ for each $v \in V$
$\mathsf{M}' \leftarrow \emptyset$
while $F \neq \emptyset$ **do**
  solve **LP-M**$[W, F, \mathcal{M}', b']$ to get an optimum extreme point x^*
  if $x^*_e = 0$ for some $e \in F$ **then**
    $F \leftarrow F - \{e\}$
    $\mathcal{M}' \leftarrow \mathcal{M}' - e$ ▷ fix x^*_e to 0 from now on
  else if $x^*_e = 1$ for some $e \in F$ **then**
    $F \leftarrow F - \{e\}$
    $\mathcal{M}' \leftarrow \mathcal{M}'/e$
    $\mathsf{M}' \leftarrow \mathsf{M}' \cup \{e\}$ ▷ fix x^*_e to 1 from now on
    $b'_v \leftarrow b'_v - d_{v,e}$ for each endpoint v of e ▷ permanently allocate space for e
  else
    let v be any vertex in W with minimum value $|\delta_F(v)| - x^*(\delta_F(v))$
    $W \leftarrow W - \{v\}$ ▷ drop the capacity constraint for v
return M'

Proof. Let $A_1 \subsetneq A_2 \subsetneq \ldots \subsetneq A_t \subseteq F$ be any *maximal-length chain of tight sets*. That is, $x^*(A_i) = r_{\mathcal{M}'}(A_i)$ for each A_i in the chain. Then the indicator vectors $\chi_{A_i} \in \{0,1\}^F$ of the sets A_i are linearly independent and every other $A \subseteq F$ with $x^*(A_i) = r_{\mathcal{M}'}(A)$ has $\chi_A \in \text{span}\{\chi_{A_i} : 1 \leq i \leq t\}$. This can be proven by using uncrossing techniques that exploit submodularity of $r_{\mathcal{M}'}$, see Chapter 5 of [10].

Now, as $A_{i-1} \subsetneq A_i$ for $1 < i \leq t$ and $x^*_e > 0$ for each $e \in F$, we see $r_{\mathcal{M}'}(A_i) = x^*(A_i) > x^*(A_{i-1}) = r_{\mathcal{M}'}(A_{i-1})$. Since the ranks are integral and $r_{\mathcal{M}'}(A_1) \neq 0$ (as $A_1 \neq \emptyset$), then $r_{\mathcal{M}}(A_i) \geq i$ for all $1 \leq i \leq t$.

Note that $|F| \leq t + |W|$ because of the number of non-zero (fractional) variables is at most the size of a basis for the tight constraints. We have

$$\begin{aligned}\sum_{v \in W} |\delta_F(v)| - x^*(\delta_F(v)) &\leq \sum_{v \in V} |\delta_F(v)| - x^*(\delta_F(v)) \leq k \cdot (|F| - x^*(F)) \\ &\leq k \cdot (|F| - r_{\mathcal{M}'}(A_t)) \leq k \cdot (|F| - t) \leq k \cdot |W|.\end{aligned}$$

The second bound holds because each edge has at most k endpoints, so it can contribute $1 - x^*_e \geq 0$ at most k times throughout the sum. Thus, some $v \in W$ satisfies the claim. ◀

▶ **Lemma 9.** *Algorithm 1 returns a set $\mathsf{M}' \in \mathcal{I}$ such that $d_v(\delta_{\mathsf{M}'}(v) - L(v)) \leq b_v$ where $L(v)$ denotes the $\min\{k, |\delta_{\mathsf{M}'}(v)|\}$ edges $e \in \delta_{\mathsf{M}'}(v)$ with greatest demand $d_{v,e}$ across v.*

Proof. We know $\mathsf{M}' \in \mathcal{I}$ by Lemma 7. Consider an iteration where a vertex $v \in W$ is removed from W. Claim 8 shows $|\delta_F(v)| \leq x^*(\delta_F(v)) + k$.

Let $F^k_v = \{e_1, \ldots, e_k\}$ be the k edges of this iteration in $\delta_F(v)$ having largest demand (if $|\delta_F(v)| < k$ then let $F^k_v = \delta_F(v)$). Then

$$\sum_{e \in \delta_F(v) - F^k_v} d_{v,e} \leq \sum_{e \in \delta_F(v)} d_{v,e} \cdot x^*_{e,v} \leq b'_v.$$

The first bound follows because if we shift x^*-values from larger- to smaller-demand edges the value $\sum_{e \in \delta_F(v)} d_{v,e} x^*_{e,v}$ does not increase. We can continue to do this until each $e \in \delta_F(v) - F^k_v$ has one unit of x^*-mass because $|\delta_F(v)| - k \leq x^*(\delta_F(v))$.

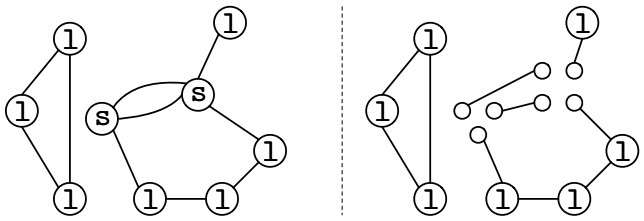


Figure 1 Left: The graph with vertex labels `s` and `l` and edges A. **Right**: The graph $\overline{G}$ obtained by "shattering" the `s` vertices. Notice the maximum degree is 2, the `ss` edges are isolated, and the `sl` edges lie on paths.

At this point of the algorithm, we have $d_v(\delta_{\mathsf{M}}(v)) = b_v - b'_v$ (letting M denote the set M' from the current iteration). So $d_v(\delta_F(v) - F_v^k) + d_v(\delta_{\mathsf{M}}(v)) \leq b_v$. We conclude by noting the edges returned by the algorithm contains only edges in $\mathsf{M} \cup F$ so $d_v(\mathsf{M}' - L(v)) \leq b_v$. ◀

Proof of Corollary 2. If there is no feasible solution to (**LP-M**), then there can be no integral solution. Otherwise, we use the same iterated relaxation technique as in Algorithm 1, except on (**LP-B**), whose polytope is the restriction of the polytope from (**LP-M**) to the base polytope of $\mathcal{M}$ (which is also integral, Corollary 40.2d of [14]).

All arguments are proven in essentially the same way. So we can find, in polynomial time, a base B with $p(\mathsf{B}) \leq \mathrm{OPT}(\textbf{LP-B})$ where $d_v(\delta_{\mathsf{B}}(v)) \leq b_v + k \cdot \max_{e \in \delta_{\mathsf{B}}(v)} d_{v,e}$. ◀

2.2 Pruning phase

We focus on $\mathsf{GDM}_{\mathsf{M}}$ ($k = 2$) in this section and show how to prune a set $\mathsf{M}' \subseteq E$ satisfying the properties of Lemma 9 to a feasible solution $\mathsf{M} \subseteq \mathsf{M}'$ while controlling the loss in its value. Each part of Theorem 1 is proved through the following lemmas. In each, for a vertex $v \in V$ we let $L(v)$ be the two edges with highest d_v-value in $\delta_{\mathsf{M}'}(v)$ (or $L(v) = \delta_{\mathsf{M}'}(v)$ if $|\delta_{\mathsf{M}'}(v)| \leq 1$). We also let $S(v) = \delta_{\mathsf{M}'}(v) - L(v)$ be the remaining edges. Note $d_v(\delta_{S(v)}(v)) \leq b_v$.

▶ **Lemma 10.** *For arbitrary graph G and arbitrary demands, we can find a feasible demand matching $\mathsf{M} \subseteq \mathsf{M}'$ with $p(\mathsf{M}) \geq p(\mathsf{M}') \cdot \frac{3}{25}$.*

Proof. For each vertex v, label v randomly with `s` with probability α or with `l` with probability $1 - \alpha$ (for α to be chosen later). Say $e \in \mathsf{M}'$ *agrees* with the labelling for an endpoint v if either $e \in S(v)$ and v is labelled `s`, or $v \in L(v)$ and v is labelled `l`. Let $\mathsf{A} \subseteq \mathsf{M}'$ be the edges agreeing with the labelling on both endpoints.

Modify the graph (V, A) by replacing each $v \in V$ labelled `s` with $|\delta_{\mathsf{A}}(v)|$ vertices and reassigning the endpoint v of each $e \in \delta_{\mathsf{A}}(v)$ to one of these vertices in a one-to-one fashion. See Figure 1 for an illustration. Call this new graph $\overline{G}$.

Each vertex in $\overline{G}$ has degree at most 2 so $\overline{G}$ decomposes naturally into paths and cycles. Each path with ≥ 2 edges can be decomposed into 2 matchings and each cycle can be decomposed into 3 matchings. Randomly choose one such matching for each path and cycle to keep and discarding the remaining edges on these paths and cycles. Note edges uv of $\overline{G}$ where u and v both had degree 1 are not discarded.

Let M be the resulting set of edges, viewed in the original graph G. Note that M is feasible: any vertex labelled `s` already had its capacity satisfied by A because $\delta_{\mathsf{A}}(v) \subseteq S(v)$. Any vertex labelled `l` has at most one of its incident edges in A chosen to stay in M.

Let $e = uv \in \mathsf{M}'$, we place a lower bound on $\mathbf{Pr}[e \in \mathsf{M}]$ by analyzing a few cases.

- If $e \in S(u) \cap S(v)$, then $\mathbf{Pr}[e \in \mathsf{M}] = \mathbf{Pr}[e \in \mathsf{A}] = \alpha^2$.

- If $e \in S(u) \cap L(v)$ or vice-versa then $\mathbf{Pr}[e \in \mathsf{M}] = \alpha \cdot (1-\alpha)/2$ (note e does not lie on a cycle in $\overline{G}$ since one endpoint is labelled $\mathtt{s}$).
- If $e \in L(u) \cap L(v)$ then $\mathbf{Pr}[e \in \mathsf{M}] = (1-\alpha)^2/3$.

Choosing $\alpha = 2/5$, we have $\mathbf{E}[p(\mathsf{M})] \geq p(\mathsf{M}') \cdot \frac{3}{25}$. ◀

We can efficiently derandomize this technique as follows. First, we use a pairwise independent family of random values to generate a probability space over labelings of V with $O(|V|)$ events such that the distribution of labels over pairs $u, v \in V$ is the same as with independently labelling the vertices. See Chapter 11 of [12] for details of this technique. For each such labelling, we decompose the paths and cycles of $\overline{G}$ into matchings and keep the most profitable matching from each path and cycle instead of randomly picking one.

▶ **Lemma 11.** *For a conflict-free instance of* $\mathsf{GDM}_{\mathsf{M}}$*, we can find a feasible solution* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{4}$.

Proof. The set A from the proof of Lemma 10 is already feasible so it does not need to be pruned further. In this case, choose $\alpha = 1/2$. ◀

▶ **Lemma 12.** *If the given graph* G *is bipartite, then we can find a feasible solution* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{7}$.

Proof. Say V_L, V_R are the two sides of V. We first partition M' into 4 groups:

$$\{uv \in \mathsf{M}' : uv \in S(u) \cap S(v)\},$$
$$\{uv \in \mathsf{M}' : uv \in L(u) \cap L(v)\},$$
$$\{uv \in \mathsf{M}' : uv \in S(u) \cap L(v)\},$$
$$\{uv \in \mathsf{M}' : uv \in L(u) \cap S(v)\}.$$

The first set is feasible. The latter three sets can each be partitioned into two feasible sets as follows. For one of these sets, form $\overline{G}$ as in the proof of Lemma 10. Each cycle can also be decomposed into two matchings because G, thus $\overline{G}$, is bipartite. Between all sets listed above, we have partitioned M' into 7 feasible sets. Let M be one with maximum profit. ◀

▶ **Lemma 13.** *For an arbitrary graph* $G = (V, E)$ *with a consistent ordering on edges, we can find a feasible demand matching* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{5}$.

Proof. We partition M' into five groups in this case. Consider the edges in decreasing order of the consistent ordering. When edge $e = uv$ is considered, assign it to a group that does not include edges in $L(u) \cup L(v)$ that come before e in the ordering. As $|L(u) \cup L(v)| \leq 4$, the edges can be partitioned into five groups this way. Each group A is a feasible demand matching since $\delta_{\mathsf{A}}(v) \subseteq S(v)$ or $|\delta_{\mathsf{A}}(v)| = 1$ for each vertex v. Now let M be the group with maximum profit, so $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{5}$. ◀

▶ **Lemma 14.** *If* $d_{v,e} \leq \epsilon \cdot b_v$ *for each* $v \in V, e \in \delta(v)$*, we can find a feasible demand matching* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq (1 - O(\epsilon^{1/3})) \cdot p(\mathsf{M}')$.

This is proven using a common randomized pruning procedure so the proof is skipped in this extended abstract. See, for example, [3] for a similar treatment in another packing problem.

3 Demand Matching in Excluded-Minor Families

In this section we prove GDM admits a PTAS in graphs that exclude a fixed graph as a minor. Our proof of Theorem 4 (the **NP**-hardness) appears in the full version. Throughout we let OPT denote the optimum solution value to the given GDM instance.

Let H be a graph and let $\mathcal{G}_H$ be all graphs that exclude H as a minor. Our PTAS uses the following decomposition.

▶ **Theorem 15** (Demaine, Hajiaghayi, and Kawarabayashi [4]). *There is a constant c_H depending only on H such that for any k and any $G \in \mathcal{G}_H$, the vertices V of G can be partitioned into $k+1$ disjoint sets so that the union of any k of these sets induce a graph with treewidth bounded by $c_H \cdot k$. Such a partition can be found in time that is polynomial in $|V|$.*

Using this decomposition, we have a PTAS for GDM when $G \in \mathcal{G}_H$ if we have a PTAS for GDM in bounded-treewidth graphs.

Intuition for our approach is given at the end of Section 1.1. We assume, for simplicity, that all $d_{v,e}$-values are distinct so we can naturally speak of *the* largest demands in a set. This is without loss of generality, we could scale demands and capacities by a common value so they are integers and then subtract $\frac{2i+j}{3|E|^2}$ from the j'th endpoint of the i'th edge according to some arbitrary ordering. Such a perturbation does not change feasibility of solutions as the total amount subtracted from all edges is < 1.

3.1 A Sparsification Lemma

We present our sparsification lemma, which even holds for general instances of $\mathsf{GDM}_{\mathsf{M}}$.

▶ **Lemma 16** (Sparsification Lemma). *For each $\epsilon > 0$ there is a feasible solution $\mathsf{M} \subseteq E$ with the following properties.*

- $p(\mathsf{M}) \geq (1-2\epsilon) \cdot \mathrm{OPT}$
- *for each $v \in V$, there is some $M_v \subseteq \mathsf{M}$ with $|M_v| \leq 1/\epsilon^2$ such that $d_{v,e} \leq \epsilon \cdot (b_v - d_v(\delta_{M_v}(v)))$ for all $e \in \delta_{\mathsf{M}-M_v}(v)$*

Think of M_v as the "large" edges in $\delta_{\mathsf{M}}(v)$ and $\delta_{\mathsf{M}-M_v}(v)$ as the "small" edges in $\delta_{\mathsf{M}}(v)$. Note that some $e \in \mathsf{M}$ may be designated large on one endpoint and small on the other.

Proof. Let M^* be an optimum solution. For each $v \in V$, if $|\delta_{\mathsf{M}^*}(v)| \geq 1/\epsilon^2$ then let L_v be the $1/\epsilon^2$ edges in $\delta_{\mathsf{M}^*}(v)$ with greatest d_v-demand and R_v be a random subset of L_v of size $1/\epsilon$. If $|\delta_{\mathsf{M}^*}(v)| < 1/\epsilon^2$, simply let $L_v = \delta_{\mathsf{M}^*}(v)$ and $R_v = \emptyset$.

Set $\mathsf{M} = \mathsf{M}^* - \cup_{v \in V} R_v$ and for each $v \in V$ set $M_v = \mathsf{M} \cap L_v$. Clearly M is feasible as it is a subset of the optimum solution. For each $e = uv \in \mathsf{M}^*$, e lies in R_u or R_v with probability at most ϵ each, so $\mathbf{Pr}[e \notin \mathsf{M}] \leq 2\epsilon$. Thus, $\mathbf{E}[p(\mathsf{M})] \geq (1-2\epsilon) \cdot \mathrm{OPT}$.

Now we focus on proving the second property for M. Let v be an arbitrary vertex in V. By construction $|M_v| \leq |L_v| \leq 1/\epsilon^2$. If $|R_v| = 0$ then $\delta_{\mathsf{M}-M_v}(v) = \emptyset$, otherwise, $|R_v| = 1/\epsilon$ and for each remaining $e \in \delta_{\mathsf{M}-M_v}(v)$, we note that $d_{v,e} + d_v(\delta_{M_v}(v)) + \sum_{e' \in R_v} d_{v,e'} \leq b_v$ because the terms represent a subset of edges of M^* incident to v. Rearranging and using the fact that $d_{v,e'} \geq d_{v,e}$ for any $e' \in R_v$ shows $\frac{1}{\epsilon} \cdot d_{v,e} \leq b_v - d_v(\delta_{M_v}(v))$. ◀

This motivates the following notion of a relaxed solution.

▶ **Definition 17.** An ϵ-relaxed solution is a subset $\mathsf{M} \subseteq E$ along with sets $M_v \subseteq \delta_{\mathsf{M}}(v)$ with $|M_v| \leq 1/\epsilon^2$ for each $v \in V$ such that the following hold. First, let $\overline{b}_v = b_v - d_v(\delta_{M_v}(v))$ for each $v \in V$. Next, for each $e \in \delta_{\mathsf{M}-M_v}(v)$, let $d'_{v,e}$ be the value of $d_{v,e}$ rounded down to the nearest integer multiple of $\frac{\epsilon}{|E|}\overline{b}_v$. Then the following must hold:

- **Large Edge Feasibility**: $d_v(\delta_{M_v}(v)) \leq b_v$ for each $v \in V$.
- **Small Edges**: $d_{v,e} \leq \epsilon \bar{b}_v$ for each $v \in V$ and each $e \in \delta_{\mathsf{M}-M_v}(v)$.
- **Discretized Small Edge Feasibility**: $d'_v(\delta_{\mathsf{M}-M_v}(v)) \leq \overline{b_v}$ for each $v \in V$.

The set M in an ϵ-relaxed solution is not necessarily a feasible GDM solution under the original demands d. As we will see shortly, it can be pruned to get a feasible solution without losing much value. Note the scaling from d to d' for some of the edges e in the definition is done independently for each endpoint of e: the demand at different endpoints may be shifted down by different amounts.

Sometimes we informally say just a set $\mathsf{M} \subseteq E$ itself is an ϵ-relaxed solution even if we do not explicitly mention the corresponding M_v sets.

▶ **Lemma 18.** *Let* M *be an* ϵ*-relaxed solution with maximum possible value* $p(\mathsf{M})$*. Then* $p(\mathsf{M}) \geq (1 - 2\epsilon) \cdot OPT$*.*

Proof. The set M and its corresponding M_v subsets from Lemma 16 suffices. ◀

▶ **Lemma 19.** *Given any* ϵ*-relaxed solution* $\mathsf{M} \subseteq E$*, we can efficiently find some* $\mathsf{M}' \subseteq \mathsf{M}$ *that is a feasible* GDM *solution with* $p(\mathsf{M}') \geq (1 - O(\epsilon^{1/3})) \cdot p(\mathsf{M})$*.*

The idea is that the $\{0, 1\}$ indicator vector of M is almost a feasible solution to (**LP-M**) with the trivial matroid $\mathcal{I} = 2^E$ in the residual instance after all "large" edges are packed so it can be pruned to a feasible solution while losing very little value by appealing to the last bound in Theorem 1. There is a minor subtlety in how to deal with edges that are both "small" and "large". The proof is deferred to the full version.

3.2 A Dynamic Programming Algorithm

Suppose $G = (V, E)$ has treewidth at most τ and that we are given a tree decomposition $\mathcal{T} = (\mathcal{B}, E_{\mathcal{T}})$ of G where each $B \in \mathcal{B}$ has $|B| \leq \tau + 1$. Recall this means the following:

1. For each $v \in V$, the set of bags $\mathcal{B}_v = \{B \in \mathcal{B} : v \in B\}$ form a connected subtree of $\mathcal{T}$.
2. For each $uv \in E$, there is at least one bag $B \in \mathcal{B}$ with $u, v \in B$.

Let $B^r \in \mathcal{B}$ be some arbitrarily chosen *root* bag. View $\mathcal{T}$ as being rooted at B^r. We may assume that each $B \in \mathcal{B}$ has at most two children. In fact, it simplifies our recurrence a bit to assume each $B \in \mathcal{B}$ is either a leaf in $\mathcal{T}$ or has precisely two children. This is without loss of generality. Arbitrarily order the children of a non-leaf vertex so one is the *left* child and one is the *right* child. For a bag B, let $\mathcal{T}_B$ be the subtree of $\mathcal{T}$ rooted at B (so $\mathcal{T}_{B^r} = \mathcal{T}$).

For each $v \in V$, say $\overline{B}_v$ is the bag containing v that is closest to the root B^r. Note for $uv \in E$ with $\overline{B}_u \neq \overline{B}_v$ that one of $\overline{B}_u$ or $\overline{B}_v$ lies on the path between the other and B^r (by the properties of tree decompositions). For each $B \in \mathcal{B}$ and each $v \in B$, we partition a subset of the edges of $\delta(v)$ into four groups:

- $\delta^{\texttt{here}}(v : B) = \{uv \in \delta(v) : \overline{B}_u = B\}$.
- $\delta^{\texttt{left}}(v : B) = \{uv \in \delta(v) : \overline{B}_u$ lies in the left subtree of $B\}$.
- $\delta^{\texttt{right}}(v : B) = \{uv \in \delta(v) : \overline{B}_u$ lies in the right subtree of $B\}$.
- $\delta^{\texttt{up}}(v : B) = \{uv \in \delta(v) : \overline{B}_u$ lies between B and $B^r\}$.

The only other edges $uv \in \delta(v)$ not accounted for here do not have $\overline{B}_u$ in either $\mathcal{T}_B$ or between B and B^r. We note if $B = \overline{B}_v$, then every edge in $\delta(v)$ lies in one of the four groups and for any $uv \in \delta^{\texttt{up}}(v : B)$ we must have $u \in B$ (otherwise no bag contains u and v, which is impossible since $uv \in E$) and, consequently, $\overline{B}_u$ lies between B and B^r. This will be helpful to remember when we describe the recurrence.

Dynamic Programming States

Let $\Delta := \{\texttt{here}, \texttt{left}, \texttt{right}, \texttt{up}\}$ be the set of "directions" used above. The DP states are given by tuples Φ with the following components.

- A bag $B \in \mathcal{B}$.
- For each $v \in B$, a subset $M_v \subseteq \delta(v)$ with $|M_v| \leq 1/\epsilon^2$.
- For each $v \in B$ and $\kappa \in \Delta$, an integer $a_{v,\kappa} \in \{0, \ldots, |E|/\epsilon\}$ such that $\sum_{\kappa \in \Delta} a_{v,\kappa} \leq \frac{|E|}{\epsilon}$.

The number of such tuples is at most $|\mathcal{B}| \cdot |E|^{O(\tau/\epsilon^2)} \cdot (|E|/\epsilon)^{O(\tau)}$, which is polynomial in G when τ and ϵ are regarded as constants. The idea behind $a_{v,\kappa}$ is that it describes how to reserve the discretized d'_v-demand for edges $uv \in \delta^\kappa(v : B) - M_v$. Of course, other edges in $\delta(v)$ not in the partitions $\delta^\kappa(v : B)$ may be in an optimal ϵ-relaxed solution. They will either be explicitly guessed in M_v or will be considered in a state higher up the tree by the time the bag $\overline{B}_v$ is processed.

Dynamic Programming Values

For each such tuple $\Phi = (B; \langle M_v \rangle_{v \in B}; \langle a_{v,\kappa} \rangle_{v \in B, \kappa \in \Delta})$, we let $f(\Phi)$ denote the maximum total value of an ϵ-relaxed solution $\mathsf{M}' \subseteq E$ (with corresponding large sets M'_v for $v \in V$) satisfying the following properties. We slightly abuse notation and say $v \in \mathcal{T}_B$ for some $v \in V$ if v lies in some bag of the subtree $\mathcal{T}_B$.

- Each $uv \in \mathsf{M}'$ has at least one endpoint in $\mathcal{T}_B$.
- $M'_v = M_v$ for each $v \in B$.
- Each $uv \in \mathsf{M}'$ with both $\overline{B}_u, \overline{B}_v \notin \mathcal{T}_B$ lies in $M'_u \cup M'_v$.
- For $v \in B$ let $\overline{b_v} = b_v - d_v(\delta_{M'_v}(v))$. For $\kappa \in \Delta$ and $v \in B$, it must be that $d'_v(\delta^\kappa(v : B) \cap \mathsf{M}' - M'_v) \leq a_{v,\kappa} \cdot \frac{\epsilon}{|E|} \cdot \overline{b_v}$ where $d'_{v,e}$ is the largest integer multiple of $\frac{\epsilon}{|E|} \cdot \overline{b_v}$ that is at most $d_{v,e}$ for $e \in \delta_{\mathsf{M}' - M'_v}(v)$.

The last point is a bit technical. Intuitively, it says the scaled demand of small edges incident to v coming from some direction $\kappa \in \Delta$ fit in the capacity of v reserved for that direction.

If there is no such F, we say $f(\Phi) = -\infty$. Note the maximum of $f(\Phi)$ over all configurations Φ for the root bag B_r is the maximum value over all ϵ-relaxed solutions.

3.2.1 The Recurrence

For the sake of space, details behind the recurrence are deferred to the full version. We just outline the main ideas. A tuple Φ is a base case if the bag B is a leaf of $\mathcal{T}$. In this case, only edges in some $\delta^\kappa(v : B)$ set for $\kappa \in \{\texttt{here}, \texttt{up}\}$ are considered (there are none in the directions `left`, `right`). We find the optimal way to pack such edges that are not part of a "large" set M_v while ensuring the d'_v-demands do not violate the residual capacities $\overline{b_v}$ and, in particular, for each direction κ we ensure this packing does not violate the part of the residual capacity for that direction allocated by the $a_{v,\kappa}$ values. This subproblem is just the Multi-Dimensional Knapsack problem with $2|B|$ knapsacks. A standard pseudopolynomial-time algorithm can be used to solve it as the scaled demands are from a polynomial-size discrete range.

For the recursive step, we try all pairs of configurations $\Phi^{\texttt{left}}, \Phi^{\texttt{right}}$ that are "consistent" with Φ. Really this just means they agree on the sets M_v for shared vertices v and they agree on how much demand $a_{v,\kappa}$ should be allocated for each direction. For each such consistent pair, we pack small edges in $\delta^{\texttt{here}}(v : B)$ and $\delta^{\texttt{up}}(v : B)$ optimally such that their scaled demands do not violate the $a_{v,\kappa}$-capacities, again using Multi-Dimensional Knapsack.

References

1. Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing $2 + \varepsilon$ approximation for unsplittable flow on a path. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 26–41. Society for Industrial and Applied Mathematics, 2014.
2. Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory of Computing*, 8(1):533–565, 2012.
3. Gruia Calinescu, Amit Chakrabarti, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms*, 7, 2011.
4. Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *Proceedings of the Forty-sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 637–646. IEEE, 2005.
5. Takuro Fukunaga and Hiroshi Nagamochi. Network design with weighted degree constraints. *Discrete Optimization*, 7(4):246–255, 2010.
6. Georgii Gens and Evgenii Levner. Complexity of approximation algorithms for combinatorial problems: a survey. *ACM SIGACT News*, 12(3):52–65, 1980.
7. Mohammad Ghodsi, Hamid Mahini, Kian Mirjalali, Shayan Oveis Gharan, Morteza Zadimoghaddam, et al. Spanning trees with minimum weighted degrees. *Information Processing Letters*, 104(3):113–116, 2007.
8. Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.
9. Madhukar Korupolu, Adam Meyerson, Rajmohan Rajaraman, and Brian Tagiku. Coupled and k-sided placements: generalizing generalized assignment. *Mathematical Programming*, 154(1-2):493–514, 2015.
10. Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.
11. Michael J. Magazine and Maw-Sheng Chern. A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 9(2):244–247, 1984.
12. Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.
13. Ojas Parekh. Iterative packing for demand and hypergraph matching. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 349–361. Springer, 2011.
14. Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.
15. Bruce Shepherd and Adrian Vetta. The demand-matching problem. *Mathematics of Operations Research*, 32(3):563–578, 2007.
16. Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 661–670. ACM, 2007.
17. Mohit Singh and Hehui Wu. Nearly tight linear programming bounds for demand matching in bipartite graphs. `http://cgi.cs.mcgill.ca/~hehui/paper/Demand_matching.pdf`, 2012.
18. David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 681–690. ACM, 2006.

Covering Vectors by Spaces: Regular Matroids*

Fedor V. Fomin[1], Petr A. Golovach[2], Daniel Lokshtanov[3], and Saket Saurabh[4]

1 **Department of Informatics, University of Bergen, Bergen, Norway**
fedor.fomin@ii.uib.no
2 **Department of Informatics, University of Bergen, Bergen, Norway**
petr.golovach@ii.uib.no
3 **Department of Informatics, University of Bergen, Bergen, Norway**
daniello@ii.uib.no
4 **Department of Informatics, University of Bergen, Bergen, Norway; and Institute of Mathematical Sciences, Chennai, India**
saket@imsc.res.in

Abstract

We consider 5the problem of covering a set of vectors of a given finite dimensional linear space (vector space) by a subspace generated by a set of vectors of minimum size. Specifically, we study the Space Cover problem, where we are given a matrix M and a subset of its columns T; the task is to find a minimum set F of columns of M disjoint with T such that that the linear span of F contains all vectors of T. This is a fundamental problem arising in different domains, such as coding theory, machine learning, and graph algorithms.

We give a parameterized algorithm with running time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$ solving this problem in the case when M is a totally unimodular matrix over rationals, where k is the size of F. In other words, we show that the problem is fixed-parameter tractable parameterized by the rank of the covering subspace. The algorithm is "asymptotically optimal" for the following reasons.

Choice of matrices: Vector matroids corresponding to totally unimodular matrices over rationals are exactly the regular matroids. It is known that for matrices corresponding to a more general class of matroids, namely, binary matroids, the problem becomes W[1]-hard being parameterized by k.

Choice of the parameter: The problem is NP-hard even if $|T| = 3$ on matrix-representations of a subclass of regular matroids, namely cographic matroids. Thus for a stronger parameterization, like by the size of T, the problem becomes intractable.

Running Time: The exponential dependence in the running time of our algorithm cannot be asymptotically improved unless Exponential Time Hypothesis (ETH) fails.

Our algorithm exploits the classical decomposition theorem of Seymour for regular matroids.

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

Keywords and phrases regular matroids, spanning set, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.56

* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959 and the Research Council of Norway via the projects "CLASSIS" and "MULTIVAL".

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 56; pp. 56:1–56:15

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

1 Introduction

We consider the fundamental problem of covering a subspace of a given finite dimensional linear space (vector space) by a set of vectors of minimum size. The input of the problem is a matrix M given together with a function w assigning a nonnegative weight to each column of M and a set T of terminal column-vectors T of M. The task is to find a minimum set of column-vectors F of M (if such a set exists) which is disjoint with T and generates a subspace containing the linear space generated by T. In other words, $T \subseteq \mathrm{span}(F)$, where $\mathrm{span}(F)$ is the linear span of F. We refer to this problem as the Space Cover problem.

The Space Cover problem encompasses various problems arising in different domains. The Minimum Distance problem in coding theory asks for a minimum dependent set of columns in a matrix over GF(2). This problem can be reduced to Space Cover by finding for each column t in matrix M a minimum set of columns in the remaining part of the matrix that cover $T = \{t\}$. The complexity of this problem was asked by Berlekamp et al. [2] and remained open for almost 30 years. It was resolved only in 1997, when Vardy showed it to be NP-complete [38]. The parameterized version of the Minimum Distance problem, namely Even Set, asks whether there is a dependent set $F \subseteq X$ of size at most k. The parameterized complexity of Even Set is a long-standing open question in the area, see e.g. [8]. In the language of matroid theory, the problem of finding a minimum dependent set is known as Matroid Girth, i.e. the problem of finding a circuit in matroid of minimum length [39]. In machine learning this problem is known as the Subspace Recovery problem [20]. This problem also generalizes the problem of computing the rank of a tensor.

For our purposes, it is convenient to rephrase the definition of the Space Cover problem in the language of matroids. Given a matrix N, let $M = (E, \mathcal{I})$ denote the matroid where the ground set E corresponds to the columns of N and $\mathcal{I}$ denote the family of subsets of linearly independent columns. This matroid is called the vector matroid corresponding to matrix N. Then for matroids, finding a subspace covering T corresponds to finding $F \subseteq E \setminus T$, $F \in \mathcal{I}$, such that $|F| \leq k$ and T is spanned by F. Let us remind that in a matroid set F spans T, denoted by $T \subseteq \mathrm{span}(F)$, if $r(F) = r(T \cup F)$. Here $r\colon 2^E \to \mathbb{N}_0$ is the rank function of M. (We use $\mathbb{N}_0$ to denote the set of nonnegative integers.)

Then Space Cover is defined as follows.

Space Cover **Parameter:** k
Input: A binary matroid $M = (E, \mathcal{I})$ given together with its matrix representation over GF(2), a weight function $w\colon E \to \mathbb{N}_0$, a set of *terminals* $T \subseteq E$, and a nonnegative integer k.
Question: Is there a set $F \subseteq E \setminus T$ with $w(F) \leq k$ such that $T \subseteq \mathrm{span}(F)$?

Since a representation of a binary matroid is given as a part of the input, we always assume that the *size* of M is $||M|| = |E|$. For regular matroids, testing matroid regularity can be done efficiently, see e.g. [37], and when the input binary matroid is regular, the requirement that the matroid is given together with its representation can be omitted.

It is known (see, e.g., [26]) that Space Cover on special classes of binary matroids, namely graphic and cographic matroids, generalizes two well-studied optimization problems on graphs, namely Steiner Tree and Multiway Cut. Both problems play fundamental roles in parameterized algorithms.

Recall that in the Steiner Forest problem we are given a (multi) graph G, a weight function $w\colon E \to \mathbb{N}$, a collection of pairs of distinct vertices $\{x_1, y_1\}, \ldots, \{x_r, y_r\}$ of G, and a nonnegative integer k. The task is to decide whether there is a set $F \subseteq E(G)$ with $w(F) \leq k$

such that for each $i \in \{1, \ldots, r\}$, graph $G[F]$ contains an (x_i, y_i)-path. To see that Steiner Forest is a special case of Space Cover, for instance $(G, w, \{x_1, y_1\}, \ldots, \{x_r, y_r\}, k)$ of Steiner Forest, we construct the following graph. For each $i \in \{1, \ldots, r\}$, we add a new edge $x_i y_i$ to G and assign an arbitrary weight to it; notice that we can create multiple edges this way. Denote by G' the obtained mulitigraph and let T be the set of added edges and let $M(G')$ be the graphic matroid associated with G'. Then a set of edges $F \subseteq E(G)$ forms a graph containing all (x_i, y_i)-paths if an only if $T \subseteq \mathrm{span}(F)$ in $M(G')$.

The special case of Steiner Forest when $x_1 = x_2 = \cdots = x_r$, i.e. when set F should form a connected subgraph spanning all demand vertices, is the Steiner Tree problem, the fundamental problem in network optimization. By the classical result of Dreyfus and Wagner [10], Steiner Tree is fixed-parameter tractable (FPT) parameterized by the number of terminals. The study of parameterized algorithms for Steiner Tree has led to the design of important techniques, such as Fast Subset Convolution [3] and the use of branching walks [29]. Research on the parameterized complexity of Steiner Tree is still on-going, with recent significant advances for the planar version of the problem [33]. Algorithms for Steiner Tree are frequently used as a subroutine in FPT algorithms for other problems; examples include vertex cover problems [19], near-perfect phylogenetic tree reconstruction [4], and connectivity augmentation problems [1].

The dual of Space Cover, i.e., the variant of Space Cover asking whether there is a set $F \subseteq E \setminus T$ with $w(F) \leq k$ such that $T \subseteq \mathrm{span}(F)$ in the dual matroid M^*, is equivalent to the Restricted Subset Feedback Set problem. In this problem the task is for a given matroid M, a weight function $w\colon E \to \mathbb{N}_0$, and a nonnegative integer k, to decide whether there is a set $F \subseteq E \setminus T$ with $w(F) \leq k$ such that matroid M' obtained from M by deleting the elements of F has no circuit containing an element of T. Hence, Space Cover for cographic matroids is equivalent to Restricted Subset Feedback Set for graphic matroids. Restricted Subset Feedback Set for graphs was introduced by Xiao and Nagamochi [40], who showed that this problem is FPT parameterized by $|F|$. Let us note that in order to obtain an algorithm for Space Cover with a single-exponential dependence in k, we also need to design a new algorithm for Space Cover on cographic matroids which improves significantly the running time achieved by Xiao and Nagamochi [40].

Multiway Cut, another fundamental graph problem, is the special case of Restricted Subset Feedback Set, and therefore of Space Cover. In the Multiway Cut problem we are given a (multi) graph G, a weight function $w\colon E \to \mathbb{N}$, a set $S \subseteq V(G)$, and a nonnegative integer k. The task is to decide whether there is a set $F \subseteq E(G)$ with $w(F) \leq k$ such that the vertices of S are in distinct connected components of the graph obtained from G by deleting edges of F. Indeed, let (G, w, S, k) be an instance of Multiway Cut. We construct graph G' by adding a new vertex u and connecting it to the vertices of S. Denote by T the set of added edges and assign weights to them arbitrarily. Then (G, w, S, k) is equivalent to the instance $(M(G'), w, T, k)$ of Restricted Subset Feedback Set. If $|S| = 2$, Multiway Cut is exactly the classical min-cut problem which is solvable in polynomial time. However, as it was proved by Dahlhaus et al. [6] already for three terminals the problem becomes NP-hard. Marx, in his celebrated work on important separators [28], has shown that Multiway Cut is FPT when parameterized by the size of the cut $|F|$.

While Steiner Tree is FPT parameterized by the number of terminal vertices, the hardness results for Multiway Cut with three terminals yields that Space Cover parameterized by the size of the terminal set T is Para-NP-complete even if restricted to cographic matroids. This explains why we parameterize Space Cover by the rank of the span and not the size of the terminal set.

There is also a strong argument that Space Cover is not tractable in its full generality on binary matroids for the following reason. It follows from the result of Downey et al. [9] on the hardness of the Maximum-Likelihood Decoding problem, that Space Cover is W[1]-hard for binary matroids when parameterized by k even if restricted to the inputs with one terminal and unit-weight elements. However, it is still possible to establish the tractability of the problem on a large class of binary matroids. Sandwiched between graphic and cographic (where the problem is FPT) and binary matroids (where the problem is intractable) is the class of regular matroids. Our main theorem establishes the tractability of Space Cover on regular matroids.

▶ **Theorem 1.** Space Cover *on regular matroids is solvable in time* $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.

We believe that due to the generality of Space Cover, Theorem 1 will be useful in the study of various optimization problems on regular matroids. As an example, we consider the Rank h-Reduction problem, see e.g. [24]. Here we are given a binary matroid M and positive integers h and k, the task is to decide whether it is possible to decrease the rank of M by at least h by deleting k elements. For graphic matroids, this is the h-Way Cut problem, which is for a connected graph G and positive integers h and k, to decide whether it is possible to separate G into at least h connected components by deleting at most k edges. By the celebrated result of Kawarabayashi and Thorup [25], h-Way Cut is FPT parameterized by k even if h is a part of the input. The result of Kawarabayashi and Thorup cannot be extended to cographic matroids; we show that for cographic matroids the problem is W[1]-hard when parameterized by $h + k$. On the other hand, by making use of Theorem 1, we solve Rank h-Reduction in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(h)}$ on regular matroids.

Let us also remark that the running time of our algorithm is asymptotically optimal: unless Exponential Time Hypothesis fails, there is no algorithm of running time $2^{o(k)} \cdot ||M||^{\mathcal{O}(1)}$ solving Space Cover on graphic (Steiner Tree) or cographic (Multiway Cut) matroids, see e.g. [5].

Related work. The main building block of our algorithm is the fundamental theorem of Seymour [34] on a decomposition of regular matroids. Roughly speaking (we define it properly in Section 2), the Seymour's decomposition provides a way to decompose a regular matroid into much simpler *base* matroids that are graphic, cographic or have a constant size in such way that all "communication" between base matroids is limited to "cuts" of small rank (we refer to the monograph of Truemper [37] and the survey of Seymour [35] for the introduction to matroid decompositions). This theorem has a number of important combinatorial and algorithmic applications. Among the classic algorithmic applications of Seymour's decomposition are the polynomial time algorithms of Truemper [36] (see also [37]) for finding maximum flows and shortest routes and the polynomial algorithm of Golynski and Horton [18] for constructing a minimum cycle basis. More recent applications of Seymour's decomposition can be found in approximation, on-line and parameterized algorithms. Godberg and Jerrum [17] used Seymour's decomposition theorem for obtaining a fully polynomial randomized approximation scheme (FPRAS) for the partition function of the ferromagnetic Ising model on regular matroids. Dinitz and Kortsarz in [7] applied the decomposition theorem for the Matroid Secretary problem. In [12], Gavenciak, Král and Oum initiated the study of the Minimum Spanning Circuit problem for matroids that generalizes the classical Cycle Through Elements problem for graphs. The problem asks for a matroid M, a set $T \subseteq E$ and a nonnegative integer ℓ, whether there is a circuit C of M with $T \subseteq C$ of size at most ℓ. Gavenciak, Král and Oum [12] proved that the problem is

FPT when parameterized by ℓ if $|T| \leq 2$. Very recently, in [11], we extended this result by showing that Minimum Spanning Circuit is FPT when parameterized by $k = \ell - |T|$.

On a very superficial level, all the algorithmic approaches based on the Seymour's decomposition theorem utilize the same idea: solve the problem on base matroids and then "glue" solutions into a global solution. However, such a view is a strong oversimplification. First of all, the original decomposition of Seymour in [34] was not meant for algorithmic purposes and almost every time to use it algorithmically one has to apply nontrivial adjustments to the original decomposition. For example, in order to solve Matroid Secretary on regular matroids, Dinitz and Kortsarz in [7] had to give a refined decomposition theorem suitable for their algorithmic needs. Similarly, in order to use the decomposition theorem for approximation algorithms, Goldberg and Jerrum in [17] had to add several new ingredients to the original Seymour's construction. We face exactly the same nature of difficulties in using Seymour's decomposition theorem. Our starting point is the variant of the decomposition theorem proved by Dinitz and Kortsarz in [7]. However, this theorem as it is can also not be used "statically" for our purposes. Our algorithm, while recursively constructing a solution has to "dynamically" transform the decomposition. This occurs when the algorithm processes cographic matroids "glued" with other matroids and for that part of the algorithm the transformation of the decomposition is essential.

2 Algorithm roadmap

In this section we give a high level overview of our algorithm for Space Cover. Due to space restrictions, all details and proofs are postponed for a journal version of our paper. We assume that the reader is acquainted with the basics of Matroid theory and refer to the book of Oxley [32] for the introduction.

We denote the ground set of matroid $M = (E, \mathcal{I})$ by $E(M)$ or simply by E if it does not create confusion. Recall that a set $X \subseteq E$ *spans* $e \in E$ if $r(X \cup \{e\}) = r(X)$, and $\mathrm{span}(X) = \{e \in E \mid X \text{ spans } e\}$, where r is the rank function of M. Respectively, X *spans a set* $T \subseteq E$ if $T \subseteq \mathrm{span}(X)$. An (inclusion) minimal dependent set is called a *circuit* of M. An one-element circuit is called *loop*, and if $\{e_1, e_2\}$ is a two-element circuit, then it is said that e_1 and e_2 are *parallel*. A set $X \subseteq E$ is a *cycle* of M if X is either empty or X is a disjoint union of circuits. Let G be a (multi) graph. The *cycle* matroid $M(G)$ has the ground set $E(G)$ and a set $X \subseteq E(G)$ is independent if $X = \emptyset$ or $G[X]$ has no cycles. Notice that C is a circuit of $M(G)$ if and only if C induces a cycle of G. The *bond* matroid $M^*(G)$ with the ground set $E(G)$ is dual to $M(G)$, and X is a circuit of $M^*(G)$ if and only if X is a minimal cut-set of G. It is said that M is a *graphic* matroid if M is isomorphic to $M(G)$ for some graph G and M is *cographic* if M is isomorphic to $M^*(G)$.

Our algorithm uses the following observation.

▶ **Observation 2.** *Let $e \in E$ and $X \subseteq E \setminus \{e\}$ for a matroid M. Then $e \in \mathrm{span}(X)$ if and only if there is a circuit C such that $e \in C \subseteq X \cup \{e\}$.*

By Observation 2, to solve Space Cover we have to find $F \subseteq E \setminus T$ with $w(F) \leq k$ such that for every $t \in T$, there is circuit C of M such that $t \in C \subseteq F \cup \{t\}$.

2.1 Regular matroid decompositions

In this section we describe matroid decomposition theorems that are pivotal for the algorithm for Space Cover. Roughly speaking, the classical theorem of Seymour [34] says that every regular matroid can be decomposed via "small sums" into basic matroids which are graphic,

cographic and very special matroid of constant size called R_{10}. To describe the decomposition of matroids, we need the notion of "ℓ-sums" of matroids; we refer to [32, 37] for a formal introduction to matroid sums. However, for our purpose, it is sufficient that we restrict ourselves to binary matroids and up to 3-sums [34]. Recall that, for two sets X and Y, $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$ denotes the symmetric difference of X and Y. For two binary matroids M_1 and M_2, the *sum* of M_1 and M_2, denoted by $M_1 \triangle M_2$, is the matroid M with the ground set $E(M_1) \triangle E(M_2)$ whose cycles are all subsets $C \subseteq E(M_1) \triangle E(M_2)$ of the form $C = C_1 \triangle C_2$, where C_1 is a cycle of M_1 and C_2 is a cycle of M_2.

▶ **Definition 3** ($\{1,2,3\}$-sum)**.** For matroids M_1, M_2 and their sum M,

(S1) If $E(M_1) \cap E(M_2) = \emptyset$ and $E(M_1), E(M_2) \neq \emptyset$, then M is the 1-*sum* of M_1 and M_2 and we write $M = M_1 \oplus_1 M_2$.

(S2) If $|E(M_1) \cap E(M_2)| = 1$, then M is the 2-*sum* of M_1 and M_2 and we write $M = M_1 \oplus_2 M_2$.

(S3) If $|E(M_1) \cap E(M_2)| = 3$, the 3-element set $Z = E(M_1) \cap E(M_2)$ is a circuit of M_1 and M_2, then M is the 3-*sum* of M_1 and M_2 and we write $M = M_1 \oplus_3 M_2$.

If $M = M_1 \oplus_k M_2$ for some $k \in \{1,2,3\}$, then we write $M = M_1 \oplus M_2$.

Note that the definitions of (S2) and (S3) in [34] include some additional restrictions for $E(M_1) \cap E(M_2)$ but, as it was pointed by Dinitz and Kortsarz in [7], they are used only to ensure the nontriviality and can be omitted for algorithmic applications.

▶ **Definition 4** ($\{1,2,3\}$-decomposition)**.** A $\{1,2,3\}$-*decomposition* of a matroid M is a collection of matroids $\mathcal{M}$, called the *basic matroids*, and a rooted binary tree T in which M is the root and the elements of $\mathcal{M}$ are the leaves such that any internal node is either 1-, 2- or 3-sum of its children.

We also need the special binary matroid R_{10} which is represented over GF(2) by the 5×10-matrix whose columns are formed by vectors that have exactly three non-zero entries (or rather three ones) and no two columns are identical. Seymour's theorem [34] states that every regular matroid has a $\{1,2,3\}$-decomposition in which every basic matroid is graphic, cographic or isomorphic to R_{10}.

Dinitz and Kortsarz in [7] obtained a variant of matroid decomposition which is more handy for our purposes. This variant is based on the notion conflict graph.

▶ **Definition 5** ([7])**.** Let $(T, \mathcal{M})$ be a $\{1,2,3\}$-decomposition of a matroid M. The *conflict* (or *intersection*) graph of $(T, \mathcal{M})$ is the graph G_T with the vertex set $\mathcal{M}$ such that distinct $M_1, M_2 \in \mathcal{M}$ are adjacent in G_T if and only if $E(M_1) \cap E(M_2) \neq \emptyset$.

▶ **Theorem 6** ([7])**.** *For a given regular matroid M, there is a (conflict) tree $\mathcal{T}$, whose set of nodes is a set of matroids $\mathcal{M}$, where each element of $\mathcal{M}$ is a graphic or cographic matroid, or a matroid obtained from R_{10} by (possibly) adding parallel elements, that has the following properties: (i) if two distinct matroids $M_1, M_2 \in \mathcal{M}$ have nonempty intersection, then M_1 and M_2 are adjacent in $\mathcal{T}$, (ii) for any distinct $M_1, M_2 \in \mathcal{M}$, the intersection $E(M_1) \cap E(M_2)$ satisfies one of the properties (S1)–(S3) of 1, 2 or 3-sums, (iii) M is obtained by the consecutive performing 1, 2 or 3-sums for adjacent matroids in any order. Moreover, $\mathcal{T}$ can be constructed in a polynomial time.*

If $\mathcal{T}$ is a conflict tree for matroid M, we say that M is defined by $\mathcal{T}$.

2.2 Elementary reductions for Space Cover

In this section we give some elementary reduction rules that we apply on the instances of Space Cover to make it more structured. This structure will be exploited by our FPT algorithm. In particular, our algorithm crucially utilizes the fact that the solution F we are seeking is of size at most k. However, the way our algorithm is designed, in certain cases the weights of elements can be changed and it can occur that some elements could have been assigned weight zero by w. In this case a solution F of weight at most k does not imply that it is a solution of size at most k. These reduction rules allow us to take care of such situations.

▶ **Reduction Rule 1** (Zero-element). *If there is an element $e \in E \setminus T$ with $w(e) = 0$, then contract e.*

▶ **Reduction Rule 2** (Terminal circuit). *If there is a circuit $C \subseteq T$, then delete an arbitrary element $e \in C$ from M.*

Let us note that Reduction Rules 1 and 2 can be applied in time polynomial in $||M||$.

2.3 Solving Space Cover for basic matroids

We start by solving Space Cover on basic matroids that are building blocks of regular matroid: R_{10}, graphic and cographic matroids. For R_{10} the solution is trivial and for graphic matroids it is an easy extension of the classic Dreyfus-Wagner algorithm [10] for Steiner Tree. However, a single-exponential algorithm for cographic matroids requires new ideas.

Thus we obtain the following lemmata.

▶ **Lemma 7.** Space Cover *can be solved in polynomial time for matroids that can be obtained from R_{10} by adding parallel elements, element deletions and contractions.*

▶ **Lemma 8.** Space Cover *on graphic matroids is solvable in time $4^k \cdot ||M||^{\mathcal{O}(1)}$.*

▶ **Lemma 9.** Space Cover *can be solved in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$ on cographic matroids.*

By the results of Xiao and Nagamochi [40], Restricted Subset Feedback Set can be solved in time $2^{\mathcal{O}(k \log k)} \cdot ||M||^{\mathcal{O}(1)}$ on graphic matroids. It immediately implies that Space Cover can be solved in the same time on cographic matroids by the duality of these problems. To improve this running time and get a single-exponential dependence in k, we construct a new algorithm based on the idea of enumeration of *important cuts* proposed by Marx in [28], see also [5]. Let G be a graph such that M is isomorphic to the bond matroid $M^*(G)$ of G. By the duality of Space Cover and Restricted Subset Feedback Set, a set $F \subseteq E(G) \setminus T$ spans T if and only if the edges of T are the bridges of $G - F$. The set of circuits of M is the set of inclusion-minimal edge cut-sets of G. Hence we restate Space Cover as a cut problem in G: for a given set $T \subseteq E(G)$, we need to find a minimum set $F \subseteq E(G) \setminus T$ such that the edges of T are bridges of $G - F$. For our purpose, we need to modify the definition of an important cut given by Marx [28, 5]. Let $s \in V(G)$ be a vertex of G, $T \subseteq V(G) \setminus \{s\}$ be a set of terminals, and k be a nonnegative integer. We say that a set $W \subseteq V(G)$ is *interesting* if $G[W]$ is connected, $s \in W$, and $|T \cap W| \leq 1$. For $W \subseteq V(G)$, by $\Delta(W)$ we denote the set of edges of G with exactly one end-vertex in W. Given two interesting sets W_1 and W_2 we say that W_1 is *better than* W_2 and denote by $W_2 \preceq W_1$ if $W_2 \subseteq W_1$, $|\Delta(W_1)| \leq |\Delta(W_2)|$, and $T \cap W_1 \subseteq T \cap W_2$. For set of terminals $T \subseteq V(G) \setminus \{s\}$, an interesting set W is (s, T, k)-*semi-important* if $|\Delta(W)| \leq k$ and there is

no set W' such that $W \preceq W'$. The set of edges $\Delta(W)$ of a (s,T,k)-semi-important set W is called a (s,T,k)-*semi-important cut*. We show that the number of (s,T,k)-semi-important cuts is in $16^k \cdot n^{\mathcal{O}(1)}$. Moreover, such cuts can be enumerated within the same time. The crux in the proof of Lemma 9 is the recursive algorithm computing the solution. The running time of the algorithm can be estimated by a polynomial of the number of (s,T,k)-semi-important cuts.

2.4 Solving Space Cover for regular matroids

Now we conjure all that have developed so far and design an algorithm for Space Cover on regular matroids, running in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$. We first give some generic steps, followed by steps when matroid in consideration is either graphic and cographic and ending with a result that ties them all.

Let (M,w,T,k) be a given instance of Space Cover. First, we exhaustively apply Reduction Rules 1-2. To simplify notations, we also denote the reduced instance by (M,w,T,k). We say that a matroid M is *basic* if it is graphic, cographic or can be obtained from R_{10} by adding parallel elements. By Lemmata 7, 8, and 9, we have the following lemma.

▶ **Lemma 10.** *Let (M,w,T,k) be an instance of* Space Cover*. If M is a basic matroid, then* Space Cover *can be solved in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.*

From now onwards we assume that matroid M in the instance (M,w,T,k) is not basic. Now using Theorem 6, we find a conflict tree $\mathcal{T}$. Recall that the set of nodes of $\mathcal{T}$ is the collection of basic matroids $\mathcal{M}$, its the edges correspond to 1-, 2− and 3-sums and that M can be constructed from $\mathcal{M}$ by performing the sums corresponding to the edges of $\mathcal{T}$ in an arbitrary order. Our algorithm is based on performing *bottom-up* traversal of the tree $\mathcal{T}$. We select an arbitrarily *node r as the root* of $\mathcal{T}$. This defines the natural parent-child relationship for the nodes of $\mathcal{T}$. We say that node M_s is a *sub-leaf* if all its children are leaves of $\mathcal{T}$. Observe that there always exists a sub-leaf in a tree on at least two nodes and that this node can be found in polynomial time.

We first modify the decomposition by an exhaustive application of the following rule.

▶ **Reduction Rule 3** (Terminal flipping)**.** *If there is a child M_ℓ of a sub-leaf M_s such that there is $e \in E(M_s) \cap E(M_\ell)$ that is parallel to a terminal $t \in E(M_\ell) \cap T$ in M_ℓ, then delete t from M_ℓ and add t to M_s as an element parallel to e.*

It is easy to show that Reduction Rule 3 is safe and can be applied in polynomial time. From now we assume that there is no child M_ℓ of M_s such that there exists an element $e \in E(M_s) \cap E(M_\ell)$ that is parallel to a terminal $t \in E(M_\ell) \cap T$ in M_ℓ. This is important because it allows us to reduce the parameter while branching. In what follows, we do a bottom-up traversal of $\mathcal{T}$ and at each step we delete one of the children of M_s. A child of M_s is deleted either because of an application of a reduction rule, or because of recursively solving the problem on a smaller sized tree. It is possible that, while recursively solving the problem, we could possibly modify (or replace) M_s to encode some auxiliary information that we have already computed while solving the problem. If at some moment we arrive at the case $T = \emptyset$, then algorithm returns yes and stops. If at some moment the situation with $E \setminus T = \emptyset$ or $|T| > k$ occurs, then we return no and stop.

2.4.1 Processing leaves

For a sub-leaf node M_s, we say that a child M_ℓ of M_s is a 1, 2 or 3-*leaf* if the edge between M_s and M_ℓ corresponds to 1, 2 or 3-sum respectively. While the cases with 1- and 2-leaves

are relatively easy to settle, the case when M_ℓ is a 3-leaf is difficult. We start from generic steps which do not depend on the types of M_s and its child.

If M_ℓ is an 1-leaf such that $E(M_\ell)$ does not contain terminals, we simply delete M_ℓ from $\mathcal{T}$ and consider the problem for the matroid defined by the obtained tree. If M_ℓ is an 1-leaf such that $T_\ell = E(M_\ell) \cap T \neq \emptyset$, then we can solve Space Cover for M_ℓ with the set of terminals T_ℓ independently. More formally, we find minimum $k' \leq k$ such that $(M_\ell, w_\ell, T_\ell, k')$ is a yes-instance of Space Cover using Lemma 10. Then if such k' exist, we consider the matroid M' defined by $\mathcal{T}'$ obtained from $\mathcal{T}$ by the deletion of M_s and then solve the problem for $(M', w, T \cap E(M'), k - k')$. Safeness of this reduction immediately follows from the definition of 1-sum, and the reduction can be done in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.

For 2-leaves, we either reduce a leaf or apply a recursive procedure based on whether the leaf contains a terminal or not. Let M_ℓ be 2-leaf that is adjacent to M_s is $\mathcal{T}$ and $E(M_s) \cap E(M_\ell) = \{e\}$. Let also M' be the matroid defined by $\mathcal{T}'$ obtained from $\mathcal{T}$ by the deletion of M_ℓ.

If M_ℓ does not contain terminals, we find a circuit of M_ℓ of minimum weight w_ℓ containing e assuming that the weight of e is 0. Notice, that this can be done in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$ by solving Space Cover on M_ℓ for the unique terminal e. Then we delete M_ℓ from $\mathcal{T}$, assign the weight w_ℓ to the element e of M_s and then solve the problem for M'.

Suppose that M_ℓ is a 2-leaf with terminals. Let $T_\ell = E(M_\ell) \cap T$ and $T' = T \setminus T_\ell$. Notice that due to Reduction Rule 3, M_ℓ has no terminal parallel to e. In particular, it can be shown that this implies that the total weight of the elements of M_ℓ in any solution is positive and this makes the branching possible, because the selection of elements of a solution in M_ℓ reduces the parameter. Notice here that we allow zero weights but all such nonterminal elements are contracted by Reduction Rule 1. We have three branching cases corresponding to the behavior of a (potential) solution F. Recall that by Observation 2, for each $t \in T$, there is a circuit C such that $t \in C \subseteq F \cup \{t\}$.

Case 1. There is $t \in T'$ and a circuit C of M such that $t \in C \subseteq F \cup \{t\}$ and C contains an element of M_ℓ. To handle this case, we consider Space Cover on M_ℓ with the terminals $T_e \cup \{e\}$, that is, we declare e to be a terminal. We find the minimum $0 < k' \leq k$ such that $(M_\ell, w, T_\ell \cup \{e\}, k')$ is a yes-instance of Space Cover using Lemma 10. Then we assign the weight 0 to e in M_s and solve the problem for $(M', w, T', k - k')$.

Case 2. There is $t \in T_\ell$ and a circuit C of M such that $t \in C \subseteq F \cup \{t\}$ and C contains an element of M'. This case is handled symmetrically to Case 1 and, respectively, we find the minimum $0 < k' \leq k$ such that (M_ℓ, w, T_ℓ, k') is a yes-instance of Space Cover where e is assumed to have the weight 0. Then we solve the problem for $(M', w, T' \cup \{e\}, k - k')$.

Case 3. None of the above cases occur, i.e., every terminal from M_ℓ is spanned by elements of M_ℓ in F and every terminal from M' is spanned by elements of M'. Then we can solve the problem independently for M_ℓ and M' assuming that the weight of e is $k + 1$ which forbids using e in a solution. We find minimum $0 < k' \leq k$ such that $(M_\ell, w_\ell, T_\ell, k')$ is a yes-instance of Space Cover and then solve the problem for $(M', w, T', k - k')$.

It is possible to show this branching is exhaustive. We show also that one call of the step (without recursive calls) can be done in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.

Suppose now that M_ℓ is a 3-leaf adjacent to M_s in $\mathcal{T}$. We again differentiate between cases when it has terminals or not. Assume that M_ℓ contains $T_\ell = E(M_\ell) \cap T \neq \emptyset$. Denote

by $Z = E(M_s) \cap E(M_\ell)$. As for 2-leaves, we observe that there are no terminals of T_ℓ that are parallel to the elements of Z in M_ℓ and we can branch according to the possible variants of the behavior of a (potential) solution F. The case analysis is technically complicated, because we have more variants of the behavior of F comparing to the case analysis for 2-leaves. Still, the majority of the cases are handled by similar arguments and we omit them here, but there is one case that makes our algorithm complicated and we briefly discuss it here.

Denote by M' the matroid defined by $\mathcal{T}'$ obtained from $\mathcal{T}$ by the deletion of M_ℓ. Suppose that there is $t \in T'$ and a circuit C of M such that $t \in C \subseteq F \cup \{t\}$ and C contains an element of M_ℓ, and there is $t' \in T_\ell$ and a circuit C' of M such that $t' \in C' \subseteq F \cup \{t'\}$ and C' contains an element of M'. Then it can be shown that there are distinct $e_i, e_j \in Z$ such that $C = C_1 \triangle C_2$, $C' = C'_1 \triangle C'_2$ where C_1, C'_1 are circuits of M_ℓ, C_2, C'_2 are circuits of M', $C_1 \cap C_2 = \{e_i\}$ and $C'_1 \cap C'_2 = \{e_j\}$. We declare $w(e_i) = w(e_j) = 0$ and let the weight of the third element of Z to be $k+1$. Then we find the minimum $0 < k' \leq k$ such that $(M_\ell, w, T_\ell \cup \{e_i\}, k')$ is a yes-instance of Space Cover and afterwards solve the problem for $(M', w, T' \cup \{e_j\}, k - k')$.

However, it can lead to the following situation. We have solutions F_ℓ and F' for $(M_\ell, w, T_\ell \cup \{e_i\}, k')$ and $(M', w, T' \cup \{e_j\}, k - k')$. Then there are circuits C_ℓ of M_ℓ and C' of M' such that $e_i \in C_\ell \subseteq F_\ell \cup \{e_i\}$ and $e_j \in C' \subseteq F' \cup \{e_j\}$. If $e_j \in C_\ell$ and $e_i \in C'$, then $(F_\ell \triangle F') \setminus Z$ is not a solution for M. To avoid this situation, we have to solve a special variant of Space Cover for $(M_\ell, w, T_\ell \cup \{e_i\}, k')$ where we put on the solution F_ℓ the additional condition that $e_i \in \mathrm{span}(F_\ell \setminus e_j)$. Due to this technicality, we also have to provide algorithms solving this version of Space Cover on basic matroids. This is done by constructing variants of the algorithm from Lemmata 8 and 9.

For this branching, we show that it is exhaustive and one call of this branching step (without recursive calls) can be done in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.

We approach the most challenging part concerning processing of 3-leaves without terminals. At this stage we can assume that $\mathcal{T}$ has only 3-leaves. The way to handle this case depends on the type of the sub-leaf M_s adjacent to a 3-leaf M_ℓ. Since M_ℓ is a 3-leaf, we have that M_s is either graphic or cographic, because R_{10} has no circuit of odd size.

Suppose that M_s is a graphic matroid. Let G be a graph such that its cycle matroid $M(G)$ is isomorphic to M_s. The algorithm that constructs a good $\{1,2,3\}$-decomposition also could be used to output the graph G. We assume that $M(G) = M_s$. The idea is to replace M_ℓ by attaching a gadget to G. Recall that the circuits of $M(G)$ are exactly the cycles of G. Since Z is a circuit of $M(G)$, the elements of Z form a cycle of G. Denote by v_1, v_2, v_3 its vertices. We modify G by adding a new vertex u and making it adjacent to v_1, v_2, v_3. Denote by G' the obtained graph. We assign weights to the edges of Z and the new edges according to the possible structure of a solution F in M_ℓ by solving auxiliary instances of Space Cover on M_ℓ. It can happen that to span T in M, we only need the property that $F \cap E(M_\ell)$ spans in M_ℓ a unique edge v_iv_j of Z. Then we find a spanning set F_ℓ of minimum weight $w_{ij} \leq k$ in M_ℓ for the terminal v_iv_j such that $v_hv_i, v_hv_j \notin F_\ell$ for $h \neq i, j$. Then we define $w(v_iv_j) = w_{ij}$ if F_ℓ exists and set $w(v_iv_j) = k+1$ otherwise. The other important possibility is that $F \cap E(M_\ell)$ spans in M_ℓ all v_iv_j for distinct $i, j \in \{1,2,3\}$. We find a spanning set F_ℓ of minimum weight $w' \leq k$ in M_ℓ for the terminal set Z. We show that it is possible to assign the weights to the edges uv_i for $i \in \{1,2,3\}$ in such a way that $w(uv_1) + w(uv_2) + w(uv_3) = w'$ and $w(uv_i) + w(uv_j) \geq w(v_iv_j)$ for $i, j \in \{1,2,3\}$ if such a solution F_ℓ exist. Otherwise, we simply set $w(uv_i) = k+1$ for $i \in \{1,2,3\}$. To complete the reduction, we consider the matroid M' defined by $\mathcal{T}'$ obtained from $\mathcal{T}$ by the deletion of M_ℓ where M_s is replaced by $M(G')$. Then we solve Space Cover for (M', w, T, k). We show that the reduction can be done in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.

The case when M_s is a cographic matroid is most challenging. Let G be a graph such that the bond matroid of G is isomorphic to M_s. Without loss of generality, we can assume that G is connected. Recall also that the circuits of the bond matroid $M^*(G)$ are exactly minimal cut-sets of G.

Isomorphism between M_s and $M^*(G)$ is not necessarily unique. We choose an isomorphism between M_s and $M^*(G)$ that is beneficial for our algorithmic purposes. Let $M_\ell^{(1)}, \ldots, M_\ell^{(s)}$ denote those leaves of the conflict tree $\mathcal{T}$ that are also the children of M_s. Let $Z_i = E(M_s) \cap E(M_\ell^{(i)})$, $i \in \{1, \ldots, s\}$. If M_s has a parent M^* in $\mathcal{T}$ and $E(M_s) \cap E(M^*) \neq \emptyset$, then let $Z^* = E(M_s) \cap E(M^*)$; we *emphasize* that Z^* may not exist. Next we define the notion of a *clean cut*.

▶ **Definition 11.** We say that $\alpha(Z_i) \subseteq E(G)$ is a *clean cut* with respect to an isomorphism $\alpha\colon M_s \to M^*(G)$, if there is a component H of $G - \alpha(Z_i)$ such that (i) H has no bridge, (ii) $E(H) \cap \alpha(Z_j) = \emptyset$ for $j \in \{1, \ldots, s\}$, and (iii) $E(H) \cap \alpha(Z^*) = \emptyset$ if Z^* exists. We call H a *clean component* of $G - \alpha(Z_i)$.

Next we show that given any isomorphism between M_s and $M^*(G)$, we can obtain another isomorphism between M_s and $M^*(G)$ with respect to which we have at least one clean component.

▶ **Lemma 12.** *There is an isomorphism $\alpha\colon M_s \to M^*(G)$ and a child $M_\ell^{(i)}$ of M_s such that $\alpha(Z_i)$ is a clean cut with respect to α. Moreover, given any arbitrary isomorphism from M_s to $M^*(G)$, one can obtain such an isomorphism and a clean cut together with a clean component in polynomial time.*

Using Lemma 12, we can always assume that we have an isomorphism of M_s to $M^*(G)$ such that for a child M_ℓ of M_s in $\mathcal{T}$, $Z = E(M_s) \cap E(M_\ell)$ is mapped to a clean cut. To simplify notations, we assume that $M_s = M^*(G)$ and Z is a clean cut with respect to this isomorphism. Denote by H the clean component. Let $Z = \{e_1, e_2, e_3\}$ and let $e_i = x_i y_i$ for $i \in \{1, 2, 3\}$, where $y_1, y_2, y_3 \in V(H)$. Notice that some y_1, y_2, y_3 can be the same.

We first handle the case when $E(H) \cap T = \emptyset$. Similarly to the case of a graphic subleaf, we replace M_ℓ by a gadget. The difference is that the gadget replaces M_s *and* H. We modify G as follows. First, we delete H. Then we construct three new pairwise adjacent vertices z_1, z_2, z_3 and make z_i adjacent to x_i for $i \in \{1, 2, 3\}$. Let G' be the obtained graph. We analyze the possible structure of a solution in H and M_ℓ, and use this information to assign weight to the edges $x_i z_i$ for $i \in \{1, 2, 3\}$ and $z_i z_j$ for $i, j \in \{1, 2, 3\}$ similarly to the case of a graphic subleaf. Finally, we consider the matroid M' defined by $\mathcal{T}'$ obtained from $\mathcal{T}$ by the deletion of M_s where M_s is replaced by $M(G')$. Then we solve Space Cover for (M', w, T, k). We prove that the reduction can be done in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.

It remains to consider the case $E(H) \cap T \neq \emptyset$. In this case, we either reduce H or recursively solve the problem on smaller H. Rather than describing these steps, we observe that we can decompose M_s further using the decomposition theorem given in [37, Chapter 8] using the cut $\{x_1y_1, x_2y_2, x_3y_3\}$. This way, we obtain a new leaf with terminals and can apply the already described rules.

Concerning the total running time, observe that we apply reduction rules either in polynomial time or in $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$ time. After each reduction rule we obtain a conflict tree with a smaller number of vertices, hence we use reductions a polynomial number of times. For each of the branching rule, in the recursive call we reduce the parameter, hence the number of nodes in the corresponding search tree is in $2^{\mathcal{O}(k)}$. Therefore the running time of the algorithm is $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(1)}$.

3 Reducing rank

In the well-known h-Way Cut problem, we are given a connected graph G and positive integers h and k, the task is to find at most k edges whose removal increases the number of connected components by at least h. The problem has a simple formulation in terms of matroids: Given a graph G and an integers k, h, find k elements of the graphical matroid of G whose removal reduces its rank by at least h. This motivated Joret and Vetta [24] to introduce the Rank h-Reduction problem on matroids. Here we define Rank h-Reduction on binary matroids.

Rank h-Reduction	**Parameter:** k
Input: A binary matroid $M = (E, \mathcal{I})$ given together with its matrix representation over GF(2) and two positive integers h and k.	
Question: Is there a set $X \subseteq E$ with $\|X\| \leq k$ such that $r(M) - r(M - X) \geq h$?	

As a corollary of Theorem 1, we show that on regular matroids Rank h-Reduction is FPT for any fixed h.

We use the following lemma.

▶ **Lemma 13.** *Let M be a binary matroid and let $k \geq h$ be positive integers. Then M has a set $X \subseteq E$ with $|X| \leq k$ such that $r(M) - r(M - X) \geq h$ if and only if there are disjoint sets $F, T \subseteq E$ such that $|T| = h$, $|F| \leq k - h$, and $T \subseteq \mathrm{span}(F)$ in M^*.*

For graphic matroids, when Rank h-Reduction is equivalent to h-Way Cut, the problem is FPT parameterized by k even if h is a part of the input [25]. Unfortunately, similar result does not hold for cographic matroids.

▶ **Proposition 14.** Rank h-Reduction *is* W[1]*-hard for cographic matroids parameterized by $h + k$.*

However, by Theorem 1, for fixed h, Rank h-Reduction is FPT parameterized by k on regular matroids.

▶ **Theorem 15.** Rank h-Reduction *can be solved in time $2^{\mathcal{O}(k)} \cdot ||M||^{\mathcal{O}(h)}$ on regular matroids.*

4 Conclusion

In this paper, we used the structural theorem of Seymour for designing parameterized algorithm for Space Cover. While structural graph theory and graph decompositions serve as the most usable tools in the design of parameterized algorithms, the applications of structural matroid theory in parameterized algorithms are limited. There is a series of papers about width-measures and decompositions of matroids (see, in articular, [21, 22, 23, 27, 30, 31] and the bibliography therein) but, apart of this specific area, we are not aware of other applications except the works Gavenciak et al. [12] and our recent work [11]. In spite of the tremendous progress in understanding the structure of matroids representable over finite fields [13, 14, 15, 16], this rich research area still remains to be explored from the perspective of parameterized complexity.

As a concrete open problem, what about the parameterized complexity of Space Cover on any proper minor-closed class of binary matroids?

References

1. Manu Basavaraju, Fedor V. Fomin, Petr A. Golovach, Pranabendu Misra, M. S. Ramanujan, and Saket Saurabh. Parameterized algorithms to preserve connectivity. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 800–811. Springer, 2014. doi:10.1007/978-3-662-43948-7_66.
2. Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Information Theory*, 24(3):384–386, 1978. doi:10.1109/TIT.1978.1055873.
3. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74. ACM, 2007.
4. Guy E. Blelloch, Kedar Dhamdhere, Eran Halperin, R. Ravi, Russell Schwartz, and Srinath Sridhar. Fixed parameter tractability of binary near-perfect phylogenetic tree reconstruction. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 667–678. Springer, 2006. doi:10.1007/11786986_58.
5. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
6. Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994. doi:10.1137/S0097539792225297.
7. Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. *SIAM J. Comput.*, 43(5):1807–1830, 2014. doi:10.1137/13094030X.
8. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
9. Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM J. Comput.*, 29(2):545–570, 1999. doi:10.1137/S0097539797323571.
10. S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.
11. Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Spanning circuits in regular matroids. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1433–1441. SIAM, 2017.
12. Tomás Gavenciak, Daniel Král, and Sang-il Oum. Deciding first order properties of matroids. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392, pages 239–250. Springer, 2012.
13. James F. Geelen, A. M. H. Gerards, and Geoff Whittle. Branch-width and well-quasi-ordering in matroids and graphs. *J. Comb. Theory, Ser. B*, 84(2):270–290, 2002. doi:10.1006/jctb.2001.2082.
14. Jim Geelen, Bert Gerards, and Geoff Whittle. Excluding a planar graph from gf(q)-representable matroids. *J. Comb. Theory, Ser. B*, 97(6):971–998, 2007. doi:10.1016/j.jctb.2007.02.005.
15. Jim Geelen, Bert Gerards, and Geoff Whittle. Solving Rota's conjecture. *Notices Amer. Math. Soc.*, 61(7):736–743, 2014. doi:10.1090/noti1139.
16. Jim Geelen, Bert Gerards, and Geoff Whittle. The Highly Connected Matroids in Minor-Closed Classes. *Ann. Comb.*, 19(1):107–123, 2015. doi:10.1007/s00026-015-0251-3.

17 Leslie Ann Goldberg and Mark Jerrum. A polynomial-time algorithm for estimating the partition function of the ferromagnetic ising model on a regular matroid. *SIAM J. Comput.*, 42(3):1132–1157, 2013. doi:10.1137/110851213.

18 Alexander Golynski and Joseph Douglas Horton. A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In *Algorithm Theory – SWAT 2002, 8th Scandinavian Workshop on Algorithm Theory, Turku, Finland, July 3-5, 2002 Proceedings*, volume 2368 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 2002.

19 Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of generalized vertex cover problems. In *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*, pages 36–48, 2005. doi:10.1007/11534273_5.

20 Moritz Hardt and Ankur Moitra. Algorithms and hardness for robust subspace recovery. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT)*, volume 30 of *JMLR Proceedings*, pages 354–375. JMLR.org, 2013.

21 Petr Hlinený. Branch-width, parse trees, and monadic second-order logic for matroids. *J. Comb. Theory, Ser. B*, 96(3):325–351, 2006. doi:10.1016/j.jctb.2005.08.005.

22 Petr Hlinený and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM J. Comput.*, 38(3):1012–1032, 2008. doi:10.1137/070685920.

23 Jisu Jeong, Eun Jung Kim, and Sang-il Oum. Constructive algorithm for path-width of matroids. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1695–1704. SIAM, 2016. doi:10.1137/1.9781611974331.ch116.

24 Gwenaël Joret and Adrian Vetta. Reducing the rank of a matroid. *Discrete Mathematics & Theoretical Computer Science*, 17(2):143–156, 2015. URL: http://www.dmtcs.org/dmtcs-ojs/index.php/dmtcs/article/view/2334.

25 Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *FOCS 2011*, pages 160–169. IEEE Computer Society, 2011.

26 Leonid G. Khachiyan, Endre Boros, Khaled M. Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. On the complexity of some enumeration problems for matroids. *SIAM J. Discrete Math.*, 19(4):966–984, 2005. doi:10.1137/S0895480103428338.

27 Daniel Král'. Decomposition width of matroids. *Discrete Applied Mathematics*, 160(6):913–923, 2012. doi:10.1016/j.dam.2011.03.016.

28 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

29 Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.

30 Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

31 Sang-il Oum and Paul D. Seymour. Testing branch-width. *J. Comb. Theory, Ser. B*, 97(3):385–393, 2007. doi:10.1016/j.jctb.2006.06.006.

32 James Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, Oxford, second edition, 2011. doi:10.1093/acprof:oso/9780198566946.001.0001.

33 Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 276–285. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.37.

34 Paul D. Seymour. Decomposition of regular matroids. *J. Comb. Theory, Ser. B*, 28(3):305–359, 1980. doi:10.1016/0095-8956(80)90075-1.

35 Paul D. Seymour. Matroid minors. In *Handbook of combinatorics, Vol. 1, 2*, pages 527–550. Elsevier, Amsterdam, 1995.

36 Klaus Truemper. Max-flow min-cut matroids: Polynomial testing and polynomial algorithms for maximum flow and shortest routes. *Math. Oper. Res.*, 12(1):72–96, 1987. doi:10.1287/moor.12.1.72.

37 Klaus Truemper. *Matroid decomposition*. Academic Press, 1992.

38 Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Information Theory*, 43(6):1757–1766, 1997. doi:10.1109/18.641542.

39 D. J. A. Welsh. Combinatorial problems in matroid theory. In *Combinatorial Mathematics and its Applications (Proc. Conf., Oxford, 1969)*, pages 291–306. Academic Press, London, 1971.

40 Mingyu Xiao and Hiroshi Nagamochi. An FPT algorithm for edge subset feedback edge set. *Inf. Process. Lett.*, 112(1-2):5–9, 2012. doi:10.1016/j.ipl.2011.10.007.

Linear Kernels for Edge Deletion Problems to Immersion-Closed Graph Classes*†

Archontia C. Giannopoulou[1], Michał Pilipczuk[2], Jean-Florent Raymond[3], Dimitrios M. Thilikos[4], and Marcin Wrochna[5]

1 **Technische Universität Berlin, Berlin, Germany**
archontia.giannopoulou@tu-berlin.de
2 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl
3 **Institute of Informatics, University of Warsaw, Warsaw, Poland; and AlGCo project team, CNRS, LIRMM, Montpellier, France**
jean-florent.raymond@mimuw.edu.pl
4 **AlGCo project team, CNRS, LIRMM, Montpellier, France; and Department of Mathematics, National and Kapodistrian University of Athens, Athens, Greece**
sedthilk@thilikos.info
5 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
m.wrochna@mimuw.edu.pl

Abstract

Suppose $\mathcal{F}$ is a finite family of graphs. We consider the following meta-problem, called $\mathcal{F}$-Immersion Deletion: given a graph G and an integer k, decide whether the deletion of at most k edges of G can result in a graph that does not contain any graph from $\mathcal{F}$ as an immersion. This problem is a close relative of the $\mathcal{F}$-Minor Deletion problem studied by Fomin et al. [FOCS 2012], where one deletes vertices in order to remove all minor models of graphs from $\mathcal{F}$. We prove that whenever all graphs from $\mathcal{F}$ are connected and at least one graph of $\mathcal{F}$ is planar and subcubic, then the $\mathcal{F}$-Immersion Deletion problem admits:

- a constant-factor approximation algorithm running in time $\mathcal{O}(m^3 \cdot n^3 \cdot \log m)$;
- a linear kernel that can be computed in time $\mathcal{O}(m^4 \cdot n^3 \cdot \log m)$; and
- a $\mathcal{O}(2^{\mathcal{O}(k)} + m^4 \cdot n^3 \cdot \log m)$-time fixed-parameter algorithm,

where n, m count the vertices and edges of the input graph. Our findings mirror those of Fomin et al. [FOCS 2012], who obtained similar results for $\mathcal{F}$-Minor Deletion, under the assumption that at least one graph from $\mathcal{F}$ is planar. An important difference is that we are able to obtain a *linear* kernel for $\mathcal{F}$-Immersion Deletion, while the exponent of the kernel of Fomin et al. depends heavily on the family $\mathcal{F}$. In fact, this dependence is unavoidable under plausible complexity assumptions, as proven by Giannopoulou et al. [ICALP 2015]. This reveals that the kernelization complexity of $\mathcal{F}$-Immersion Deletion is quite different than that of $\mathcal{F}$-Minor Deletion.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

* The full version of this paper can be found as an arxiv preprint [19], https://arxiv.org/abs/1609.07780.

† This work was done while A. C. Giannopoulou was holding a post-doc position at Warsaw Center of Mathematics and Computer Science and she has also been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (ERC consolidator grant DISTRUCT, agreement No 648527). Mi. Pilipczuk and M. Wrochna are supported by the Polish National Science Center grant UMO-2013/11/D/ST6/03073. J-F. Raymond is supported by the Polish National Science Center grant UMO-2013/11/N/ST6/02706. D. Thilikos is supported by project DEMOGRAPH (ANR-16-CE40-0028).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 57; pp. 57:1–57:15

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

Keywords and phrases Kernelization, Approximation, Immersion, Protrusion, Tree-cut width

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.57

1 Introduction

On the $\mathcal{F}$-Minor Deletion problem. Let us fix a finite family of graphs $\mathcal{F}$. A graph is called *$\mathcal{F}$-minor-free* if it does not contain any graph from $\mathcal{F}$ as a minor. Given a class of graphs $\mathcal{G}$, we denote by $\mathbf{obs}_{\mathrm{mn}}(\mathcal{G})$ the set of minor-minimal graphs not in $\mathcal{G}$. The celebrated Graph Minors Theorem [32] states that for $\mathcal{G}$ closed under taking minors, the set $\mathbf{obs}_{\mathrm{mn}}(\mathcal{G})$ is finite. In other words, $\mathcal{G}$ is characterized by a finite set of minor-obstructions, as $\mathcal{G}$ is exactly the class of $\mathcal{F}$-minor-free graphs, for $\mathcal{F} = \mathbf{obs}_{\mathrm{mn}}(\mathcal{G})$. Hence, studying classes of $\mathcal{F}$-minor-free graphs for finite families $\mathcal{F}$ is the same as studying general minor-closed properties of graphs.

Fomin et al. [14] performed an in-depth study of the following parameterized[1] problem, named $\mathcal{F}$-Minor Deletion[2]: *Given a graph G and an integer parameter k, decide whether one can remove at most k vertices from G to obtain an $\mathcal{F}$-minor-free graph.* By considering different families $\mathcal{F}$, the $\mathcal{F}$-Minor Deletion problem generalizes a number of concrete problems of prime importance in parameterized complexity, such as Vertex Cover, Feedback Vertex Set, or Planarization. It is easy to see that, for every fixed k, the graph class $\mathcal{G}^{\mathrm{mn}}_{k,\mathcal{F}}$ consisting of the graphs in the YES-instances (G,k) of $\mathcal{F}$-Minor Deletion, is closed under taking of minors. By the meta-algorithmic consequences of the Graph Minors series of Robertson and Seymour [32, 30], it follows (non-constructively) that $\mathcal{F}$-Minor Deletion admits an FPT-algorithm. The optimization of the running time of such FPT-algorithms for several instantiations of $\mathcal{F}$ has been a stimulating project in parameterized algorithm design. So far, it has been focused on problems generated by minor-closed graph classes.

The goal of Fomin et al. [14] was to obtain results of general nature for $\mathcal{F}$-Minor Deletion, which would explain why many concrete problems captured as its subcases are efficiently solvable using parameterized algorithms and kernelization. This has been achieved under the assumption that $\mathcal{F}$ contains at least one planar graph. More precisely, for any class $\mathcal{F}$ that contains at least one planar graph, the work of Fomin et al. [14] gives the following:

- a randomized constant-factor approximation running in time $\mathcal{O}(nm)$;
- a polynomial kernel for the problem; that is, a polynomial-time algorithm that, given an instance (G,k) of $\mathcal{F}$-Minor Deletion, outputs an equivalent instance (G',k') with $k' \leq k$ and $|G'| \leq \mathcal{O}(k^c)$, for some constant c that depends on $\mathcal{F}$;
- an FPT-algorithm for $\mathcal{F}$-Minor Deletion in time $2^{\mathcal{O}(k)} \cdot n$ (note this originally required that all graphs from $\mathcal{F}$ be connected; Kim et al. [24] showed how to lift this assumption);
- a proof that every graph in $\mathbf{obs}_{\mathrm{mn}}(\mathcal{G}^{\mathrm{mn}}_{k,\mathcal{F}})$ has at most $k^{c_{\mathcal{F}}}$ vertices, for some constant $c_{\mathcal{F}}$ that depends (non-constructively) on $\mathcal{F}$.

The assumption that $\mathcal{F}$ contains at least one planar graph is crucial for the approach of Fomin et al. [14]. Namely, from the Excluded Grid Minor Theorem of Robertson and Seymour [31] it follows that for such families $\mathcal{F}$, $\mathcal{F}$-minor-free graphs have treewidth bounded by a constant depending only of $\mathcal{F}$. Therefore, a YES-instance of $\mathcal{F}$-Minor Deletion

[1] A *parameterized* problem can be seen as a subset of $\Sigma^* \times \mathbb{N}$. For graph problems, the string x in an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ usually encodes a graph G. An FPT-algorithm for the problem is then an algorithm working in $f(k) \cdot |x|^{O(1)}$ time. See [12, 8, 29] for more on parameterized algorithms and complexity.

[2] Fomin et al. used the name $\mathcal{F}$-Deletion. We write $\mathcal{F}$-Minor Deletion instead for clarity.

roughly has to look like a constant-treewidth graph plus k additional vertices that can have arbitrary connections. Having exposed this structure, Fomin et al. [14] apply protrusion-based techniques that originate in the work on *meta-kernelization* [3, 15]. Roughly speaking, the idea is to identify large parts of the graphs that have constant treewidth and a small interface towards the rest of the graph (so-called *protrusions*), which can then be replaced by smaller gadgets with the same combinatorial behaviour. Such preprocessing based on *protrusion replacement* is the base of all three aforementioned results for $\mathcal{F}$-Minor Deletion. In the absence of a constant bound on the treewidth of an $\mathcal{F}$-minor-free graph, the technique breaks completely. In fact, the kernelization complexity of Planarization, that is, $\mathcal{F}$-Minor Deletion for $\mathcal{F} = \{K_5, K_{3,3}\}$, is a notorious open problem.

An interesting aspect of the work of Fomin et al. [14] is that the exponent of the polynomial bound on the size of the kernel for $\mathcal{F}$-Minor Deletion grows quite rapidly with the family $\mathcal{F}$. Recently, it has been shown by Giannopoulou et al. [17] that in general this growth is unavoidable: unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, for every constant η, the Treewidth-η Deletion problem (delete k vertices to get a graph of treewidth at most η) has no kernel with $\mathcal{O}(k^{\eta/4-\epsilon})$ vertices for any $\epsilon > 0$. Since graphs of treewidth η can be characterized by a finite set of forbidden minors $\mathcal{F}_\eta$, at least one of which is planar, this refutes the hypothesis that all $\mathcal{F}$-Minor Deletion problems admit polynomial kernels with a uniform bound on the degree of the polynomial. However, as shown by Giannopoulou et al. [17], such bounds can be achieved for some specific problems, like vertex deletion to graphs of constant tree-depth.

Immersion problems. Recall that a graph H can be *immersed* into a graph G (or that H is an *immersion* of G) if there is a mapping from vertices of H to pairwise different vertices of G and from edges of H to pairwise *edge-disjoint paths* connecting the images of its endpoints[3]. Such a mapping is called an *immersion model*. Just like the minor relation, the immersion relation imposes a partial order on the class of graphs. Alongside with the minor order, Robertson and Seymour [33] proved that graphs are well-quasi-ordered under the immersion order as well. This implies that for every graph class $\mathcal{G}$ that is closed under taking immersions, the set $\mathbf{obs}_{\mathrm{im}}(\mathcal{G})$ containing immersion minimal graphs that do not belong in $\mathcal{G}$, is finite (we call $\mathbf{obs}_{\mathrm{im}}(\mathcal{G})$ the *immersion obstruction set* of $\mathcal{G}$). Therefore $\mathcal{G}$ can be characterized by a finite set of forbidden immersions. The general intuition is that immersion is a containment relation that corresponds to *edge* cuts, whereas the minor relation corresponds to *vertex* cuts. Also, the natural setting for immersions is the setting of *multigraphs*. Hence, all the graphs considered in this paper may have parallel edges connecting the same pair of endpoints.

Recently, there has been a growing interest in immersion-related problems [28, 10, 16, 25, 18, 20, 36, 4, 9, 1, 21, 11] both from the combinatorial and the algorithmic point of view. Most importantly for us, Wollan proved in [36] an analogue of the Excluded Grid Minor Theorem, which relates the size of the largest wall graph that is contained in a graph as an immersion with a new graph parameter called *tree-cut width*. By a *subcubic graph* we mean a graph of maximum degree at most 3. The following theorem follows from the work of Wollan [36].

▶ **Theorem 1** ([18]). *For every planar subcubic graph H, there exists a constant a_H such that every graph not containing H as an immersion has tree-cut width bounded by a_H.*

In other words, for any family $\mathcal{F}$ of graphs that contains some planar subcubic graph, the tree-cut width of $\mathcal{F}$-immersion-free graphs is bounded by a universal constant depending

[3] In this paper we consider *weak immersions* only, as opposed to *strong immersions* where the paths are forbidden to traverse images of vertices other than the endpoints of the corresponding edge.

on $\mathcal{F}$ only. In Section 2 we discuss the precise definition of tree-cut width and how exactly Theorem 1 follows from the work of Wollan [36]. Also, note that if a family of graphs $\mathcal{F}$ does not contain any planar subcubic graph, then there is no uniform bound on the tree-cut width of $\mathcal{F}$-immersion-free graphs. Indeed, wall graphs are then $\mathcal{F}$-immersion-free, because all their immersions are planar and subcubic, and they have unbounded tree-cut width.

After the introduction of tree-cut width by Wollan [36], the new parameter gathered substantial interest from the algorithmic and combinatorial community [16, 28, 18, 25]. It seems that tree-cut width serves the same role for immersion-related problems as treewidth serves for minor-related problems and, in a sense, it can be seen as an "edge-analogue" of treewidth. In particular, given the tree-cut width bound of Theorem 1 and the general approach of Fomin et al. [14] to $\mathcal{F}$-Minor Deletion, it is natural to ask whether the same kind of results can be obtained for immersions where the edge removals are considered instead of vertex removals. More precisely, fix a finite family of graphs $\mathcal{F}$ containing some planar subcubic graph and consider the following $\mathcal{F}$-Immersion Deletion problem: given a graph G and an integer k, determine whether it is possible to delete at most k edges of G in order to obtain a graph that does not admit any graph from $\mathcal{F}$ as an immersion. Notice that when $\mathcal{F} = \{K_2\}$ and $\mathcal{F} = \{K_3\}$, the problem of computing the minimum size of such a set of edges can be solved in polynomial time, while it is NP-hard when $\mathcal{F} = \{K_4^-\}$ (see e.g. [5]).

Parallel to the case of $\mathcal{F}$-Minor Deletion, for every fixed k, the graph class $\mathcal{G}_{k,\mathcal{F}}^{\mathrm{im}}$ consisting of the graphs in the YES-instances (G, k) of $\mathcal{F}$-Immersion Deletion is closed under taking of immersions[4], therefore $\mathcal{O}_k^{\mathrm{im}} = \mathbf{obs}_{\mathrm{im}}(\mathcal{G}_{k,\mathcal{F}}^{\mathrm{im}})$ is a finite set, by the well-quasi-ordering of graphs under immersions [33]. Together with the immersion-testing algorithm of Grohe et al. [22], this implies that $\mathcal{F}$-Immersion Deletion admits (non-constructively) an FPT-algorithm. This naturally induces the parallel project of optimizing the performance of such FPT-algorithms for various instantiations of $\mathcal{F}$. More concretely, is it possible to extend the general framework of Fomin et al. [14] to obtain efficient approximation, kernelization, and FPT algorithms also for $\mathcal{F}$-Immersion Deletion? Theorem 1 suggests that the suitable analogue of the assumption from the minor setting that $\mathcal{F}$ contains a planar graph should be the assumption that at least one graph from $\mathcal{F}$ is planar and subcubic.

Our results. In this work we give a definitive positive answer to this question. The following two theorems gather our main results; for a graph G, by $|G|$ and $\|G\|$ we denote the cardinalities of the vertex and edge sets of G, respectively.

▶ **Theorem 2** (Constant factor approximation). *Let $\mathcal{F}$ be a finite family of connected graphs with at least one member being planar and subcubic. Given a graph G, in $\mathcal{O}(\|G\|^3 \log \|G\| \cdot |G|^3)$ time one can output a subset of edges $F \subseteq E(G)$ such that $G - F$ is $\mathcal{F}$-immersion-free and the size of F is at most c_{apx} times larger than the optimum size of a subset of edges with this property, for some constant c_{apx} depending on $\mathcal{F}$ only.*

The constant-factor approximation can be generalized to work when $\mathcal{F}$ contains disconnected graphs as well, using the approach of Fomin et al. [14, 13].

[4] Notice that if we consider deletion of vertices instead of edges, then the graph class $\mathcal{G}_k^{\mathrm{im}}$ is *not* closed under taking immersions (for example, in a star on 7 vertices with duplicated edges, deleting one vertex makes it K_3-immersion-free, but this 'duplicated' star immerses $2K_3$, which has no such vertex). This is the main reason why we believe that *edge* deletion gives a more suitable counterpart to $\mathcal{F}$-Minor Deletion for the case of immersions.

▶ **Theorem 3** (Linear kernelization and obstructions). *Let $\mathcal{F}$ be a finite family of connected graphs with at least one member being planar and subcubic. Given an instance (G,k) of $\mathcal{F}$-*Immersion Deletion*, in time $\mathcal{O}(\|G\|^4 \log \|G\| \cdot |G|^3)$ one can output an equivalent instance (G',k) with $\|G'\| \leq c_{\mathrm{ker}} \cdot k$, for some constant c_{ker} depending on $\mathcal{F}$ only. Moreover, every graph in $\mathcal{O}_k^{\mathrm{im}}$ has at most $c_{\mathcal{F}} \cdot k$ edges (for a constant $c_{\mathcal{F}}$ non-constructively depending on $\mathcal{F}$).*

Thus, Theorems 2 and 3 mirror the approximation and kernelization results and the obstruction bounds of Fomin et al. [14]. However, this mirroring is not exact as we show that, in the immersion setting, a stronger kernelization procedure can be designed. Namely, the size of the kernel given by Theorem 3 is *linear*, with only the multiplicative constant depending on the family $\mathcal{F}$, whereas in the minor setting, the exponent of the polynomial bound on the kernel size provably must depend on $\mathcal{F}$ (under plausible complexity assumptions). This shows that the immersion and minor settings behave quite differently and in fact stronger results can be obtained in the immersion setting. Observe that using Theorem 3 it is trivial to obtain a decision algorithm for $\mathcal{F}$-Immersion Deletion working in time $\mathcal{O}(c_{\mathrm{fpt}}^k + \|G\|^4 \log \|G\| \cdot |G|^3)$ for some constant c_{fpt} depending on $\mathcal{F}$ only: one simply computes the kernel with a linear number of edges and checks all the subsets of edges of size k.

Our techniques. Our approach to proving Theorems 2 and 3 roughly follows the general framework of protrusion replacement of Fomin et al. [14] (see also [3]). We first define protrusions suited for the problem of our interest. In fact, our protrusions can be seen as the edge-analogue of those introduced in [14] (as in [5]). A protrusion for us is simply a vertex subset X that induces an $\mathcal{F}$-immersion-free subgraph (which hence has constant tree-cut width, by Theorem 1), and has a constant number of edges to the rest of the graph. When a large protrusion is localized, it can be replaced by a smaller gadget similarly as in the work of Fomin et al. [14]. However, we need to design a new algorithm for searching for large protrusions, mostly in order to meet the condition that the exponent of the polynomial running time of the algorithm *does not depend* on $\mathcal{F}$. For this, we employ the important cuts technique of Marx [27] and the randomized contractions technique of Chitnis et al. [6]. All of these yield an algorithm that exhaustively reduces all large protrusions.

Unfortunately, exhaustive protrusion replacement is still not sufficient for a linear kernel. However, we prove that in the absence of large reducible protrusions, the only remaining obstacles are large groups of parallel edges between the same two endpoints (called *thetas*), and, more generally, large "bouquets" of constant-size graphs attached to the same pair of vertices. Without these, the graph is already bounded linearly in terms of the optimum solution size. The approximation algorithm can thus delete *all* edges except for the copies included in bouquets and thetas, reducing the optimum solution size by a constant fraction of the deleted set. It then exhaustively reduces protrusions in the remaining edges, and repeats the process until the graph is $\mathcal{F}$-immersion-free.

To obtain a linear kernel we need more work, as we do not know how to reduce bouquets and thetas directly. Instead, we apply the following strategy based on the idea of *amortization*. After reducing exhaustively all larger protrusions, we compute a constant-factor approximate solution F_{apx}. Then we analyze the structure of the graph $G - F_{\mathrm{apx}}$, which has constant tree-cut width. It appears that every bouquet (and theta) in G can be reduced up to size bounded linearly in the number of solution edges F_{apx} that "affect" it. After applying this reduction, we can still have large bouquets in the graph, but this happens only when they are affected by a large number of edges of F_{apx}. However, every edge of F_{apx} can affect only a constant number of bouquets and hence a simple amortization arguments shows that the total size of bouquets is linear in $|F_{\mathrm{apx}}|$, so also linear in terms of the optimum.

We remark that this part of the reasoning (the above amortization argument in particular) are fully new contributions of this work. These deviate significantly from arguments by Fomin et al. [14], which aimed at a obtaining a polynomial kernel only, instead of linear. Also, we remark that, contrary to the work of Fomin et al. [14], all our algorithms are deterministic.

For the second part of Theorem 3, we show that protrusion replacements can be realized as immersions of the original graph. This implies that, in the equivalent instance (G', k) produced by our kernelization algorithm, the graph G' is an immersion of G. Therefore any immersion-obstruction of $\mathcal{G}^{\mathrm{im}}_{k-1,\mathcal{F}}$ must already have a linear, in k, number of edges.

Application: immersion-closed parameters. We would like to highlight one meta-algorithmic application of our results, which was our original motivation. Suppose $\mathbf{p}$ is a graph parameter, that is, a function that maps graphs to $\mathbb{N}$. We shall say that $\mathbf{p}$ is *closed under immersion* if $\mathbf{p}(H) \leq \mathbf{p}(G)$ whenever H is an immersion of G; $\mathbf{p}$ is *closed under disjoint union* if $\mathbf{p}(G_1 \uplus G_2) = \max(\mathbf{p}(G_1), \mathbf{p}(G_2))$ for any graphs G_1, G_2; here, $\uplus$ denotes disjoint union.

For a parameter $\mathbf{p}$ and a constant r, define the $\mathbf{p}$-at-most-r Edge Deletion problem as follows: given a graph G and an integer k, determine whether at most k edges can be deleted from G to obtain a graph with the value of $\mathbf{p}$ at most r. We also define the associated parameter $\mathbf{p}_r(G) = \min\{k \mid \exists S \subseteq E(G) : |S| \leq k \wedge \mathbf{p}(G \setminus S) \leq r\}$ and $\mathcal{G}_{k,\mathbf{p}_r} = \{G \mid \mathbf{p}_r(G) \leq k\}$. Then the following meta-result can be derived from Theorems 2 and 3 and the fact that immersion is a well-quasi-order; a proof can be found in the full version of the paper.

▶ **Theorem 4.** *Let $\mathbf{p}$ be a graph parameter that is closed under immersion and under disjoint union and moreover is large on the class of walls*[5]*. Then, for every constant r, the $\mathbf{p}$-at-most-r Edge Deletion problem admits a constant-factor approximation and a linear kernel. Moreover, there is a constant c_r, depending (non-constructively) on r, such that for every k, every graph H in $\mathbf{obs}_{\mathrm{im}}(\mathcal{G}_{k,\mathbf{p}_r})$ has at most $c_r \cdot k$ edges.*

Natural parameters that satisfy the prerequisites of Theorem 4 include cutwidth, carving width, tree-cut width, and edge ranking; see e.g. [34, 35, 36, 26, 23] for more details. Theorem 4 mirrors a corollary by Fomin et al. [14] for the Treewidth-η Deletion problem asserting a constant-factor approximation, a polynomial kernel, a polynomial bound for the corresponding minor-obstruction set, and a single-exponential FPT algorithm, for every constant η.

Organization. This extended abstract focuses on sketching the proofs of Theorems 2 and 3. The proofs of statements marked with $\star$ are omitted and can be found in the full version [19].

2 Preliminaries

For a positive integer p, we denote $[p] = \{1, 2, \ldots, p\}$. A graph G is a pair $(V(G), E(G))$, where $V(G)$ is the *vertex set*, and $E(G)$ is a multiset of *edges*. Each edge connects two different vertices, called the *endpoints* of the edge (we do not allow loops). Note that there might be several edges (called *parallel edges*) between two vertices. An edge is *incident* to a vertex if it is one of its two endpoints.

We write $|G|$ for $|V(G)|$ and $\|G\|$ for $|E(G)|$ (counting edges with multiplicities). For a subset of vertices $X \subseteq V(G)$, $G[X]$ is the subgraph induced by X. For a subset of edges

[5] A graph parameter $\mathbf{p}$ is *large on a graph class $\mathcal{C}$* if $\{\mathbf{p}(G) \mid G \in \mathcal{C}\}$ is not a bounded set.

$F \subseteq E(G)$, $G - F$ denotes the graph G with all edges from F removed. For two subsets $X, Y \subseteq V(G)$, not necessarily disjoint, $E_G(X,Y)$ denotes the set of edges of $E(G)$ of the form xy for some $x \in X$ and $y \in Y$. The *boundary* of X is $\delta_G(X) = E_G(X, V(G) \setminus X)$.

Tree-cut width. A *near-partition* of a set X is a family of (possibly empty) subsets $X_1, \ldots, X_k$ of X such that $\bigcup_{i=1}^{k} X_i = X$ and $X_i \cap X_j = \emptyset$ for every $i \neq j$.

A *tree-cut decomposition* of a connected graph G is a pair $(T, \mathcal{X})$ where T is a tree and $\mathcal{X} = \{X_t : t \in V(T)\}$ is a near-partition of the vertices of G. Sets $\{X_t : t \in V(T)\}$ are called the *bags* of the decomposition. For a subset $W \subseteq V(T)$, define X_W as $\bigcup_{t \in W} X_t$. By rooting T at some vertex, we can talk about a *rooted tree-cut decomposition*. When G is disconnected, a tree-cut decomposition is a forest consisting of one tree for each connected component.

For each edge $e = uv$ of T, $T - uv$ has exactly two components which we call T_{uv} and T_{vu}, that contain u and v respectively. Since $\mathcal{X}$ is a near-partition, sets $X_{V(T_{uv})}$ and $X_{V(T_{vu})}$ form a near-partition of $V(G)$ (provided G is connected). We define the *adhesion* of an edge $e = uv$ of T, denoted $\mathsf{adh}_{\mathcal{T}}(e)$, as the set $E_G(X_{V(T_{uv})}, X_{V(T_{vu})})$. We omit the subscript if $\mathcal{T}$ is clear from the context. An adhesion is *thin* if it has at most 2 edges, *bold* otherwise.

We now move to the definition of tree-cut width. In fact, we do not give the original definition (it can be found in the full version of the paper), but we instead give an alternative definition that is easier to handle. Let G be a graph and $(T, \mathcal{X} = \{X_t : t \in V(T)\})$ be a tree-cut decomposition of G. For a node t of T, let $w(t)$ be the number of edges incident to t that have bold adhesions. The *width′* of the decomposition, denoted $\mathsf{width}'(T, \mathcal{X})$, is equal to $\max\{\max_{e \in E(T)} |\mathsf{adh}(e)|, \max_{t \in V(T)} |X_t| + w(t)\}$. The *tree-cut width′* of G, denoted $\mathsf{tctw}'(G)$, is the minimum width' of a tree-cut decomposition of G. The standard tree-cut width of G, denoted $\mathsf{tctw}(G)$, is similarly defined as the minimum width of a tree-cut decomposition of G, where width is defined slightly differently. We prove that both notions are equivalent.

▶ **Lemma 5** (⋆). *For every graph G, it holds that $\mathsf{tctw}(G) = \mathsf{tctw}'(G)$. Moreover, given a tree-cut decomposition $\mathcal{T}$ of G, it holds that $\mathsf{width}(\mathcal{T}) \leq \mathsf{width}'(\mathcal{T})$, and a tree-cut decomposition $\mathcal{T}'$ with $\mathsf{width}'(\mathcal{T}') \leq \mathsf{width}(\mathcal{T})$ can be computed in time $\mathcal{O}(\|G\| \cdot |G|^2 \cdot \mathsf{width}(\mathcal{T}))$.*

Ganian et al. [16] showed that bounded tree-cut width implies bounded treewidth. Besides, Kim et al. [25] showed that the dependency cannot be improved to subquadratic.

▶ **Lemma 6** (see [16]). *For any graph G, $\mathsf{tw}(G) \leq 2\mathsf{tctw}(G)^2 + 3\mathsf{tctw}(G)$.*

In our algorithms we need to adjust tree-cut decompositions to our needs, and hence we define the notion of a *neat tree-cut decomposition*. A neat tree-cut decomposition is a rooted tree-cut decomposition $(T, \mathcal{X})$ of a connected graph G that has the following properties:

- For every $e = uv \in E(T)$, the graphs $G[X_{V(T_{uv})}]$ and $G[X_{V(T_{vu})}]$ are connected.
- Suppose t is a node of T with parent s, such that the adhesion of st is thin. Then $E_G(X_{V(T_{ts})}, X_{V(T_{t's})}) = \emptyset$ for every sibling t' of t.

The second property was used by Ganian et al. [16] under the name *niceness*. It appears that any tree-cut decomposition can be made neat without increasing the width by much; the proof follows closely the lines of [16, Lemma 1].

▶ **Lemma 7** (⋆). *Given a tree-cut decomposition of $\mathsf{width}' \leq k$, a neat tree-cut decomposition of the same graph with $\mathsf{width}' \leq k^2 + 1$ can be constructed in time $\mathcal{O}(\|G\| \cdot |G|^2 \cdot k^2)$.*

The following result shows why neat tree-cut decompositions are useful: if a node of a neat decomposition has many neighboring nodes, then all but a constant number of them are connected to it via very simple adhesions.

▶ **Lemma 8** (⋆). *Let G be a graph with a neat tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ satisfying $\mathsf{width}'(\mathcal{T}) \leq r$. Let $t \in V(T)$. Then for all but at most $2r + 1$ of the edges e of T incident to t, $\mathsf{adh}(e)$ is thin and all of its edges have an endpoint in the bag at t.*

Finally, we need that the tree-cut width of a graph can be computed efficiently. For this, we can use the following 2-approximation algorithm of Kim et al. [25]. We remark that the width of the decomposition yielded is measured in terms of $\mathsf{width}(\cdot)$ and not $\mathsf{width}'(\cdot)$, but the decomposition can be adjusted to have also width' bounded by $2r$ using Lemma 5.

▶ **Theorem 9** (see [25]). *There is an algorithm that, given a graph G and an integer r, runs in time $2^{\mathcal{O}(r^2 \log r)} \cdot |G|^2$ and either concludes that $\mathsf{tctw}(w) > r$, or returns a tree-cut decomposition of G of width at most $2r$.*

For the whole paper we fix a finite family of graphs $\mathcal{F}$ with the following properties: all graphs of $\mathcal{F}$ are connected, and at least one is planar and subcubic. A graph G will be called $\mathcal{F}$-immersion-free, or $\mathcal{F}$-free for short, if G contains no graph from $\mathcal{F}$ as an immersion. By Theorem 1, the tree-cut width of an $\mathcal{F}$-free graph is bounded by a constant depending on $\mathcal{F}$ only, so by Lemma 6 also its treewidth is bounded by a constant. By combining this with Bodlaender's algorithm [2] and Courcelle's Theorem [7], we obtain the following:

▶ **Lemma 10** (⋆). *It can be checked in linear-time whether a given graph is $\mathcal{F}$-free.*

Moreover, by combining the bound on tree-cut width of an $\mathcal{F}$-free graph with Lemmas 5, 7 and Theorem 9, we obtain the following.

▶ **Lemma 11** (⋆). *There exists an algorithm that, given an $\mathcal{F}$-free graph G, runs in time $\mathcal{O}(\|G\| \cdot |G|^2)$ and computes a neat tree-cut decomposition $\mathcal{T}$ of G with $\mathsf{width}'(\mathcal{T}) \leq b_{\mathcal{F}}$, for some constant $b_{\mathcal{F}}$ depending on $\mathcal{F}$ only.*

For a graph G, by $\mathsf{OPT}(G)$ we denote the minimum number of edges that need to be deleted from G in order to obtain an $\mathcal{F}$-free graph.

3 Protrusions

Replacing protrusions. We now introduce the notion of a protrusion suited to the considered problem. In the sequel, we will only deal with $2b_{\mathcal{F}}$- and 2-protrusions, where $b_{\mathcal{F}}$ is the constructive bound on tctw' guaranteed by Lemma 11.

▶ **Definition 12.** An *r-protrusion* of a graph G is a set $X \subseteq V(G)$ such that $|\delta(X)| \leq r$ and $G[X]$ is $\mathcal{F}$-free.

As in [14], the base for our kernelization algorithm is *protrusion replacement*. That is, we iteratively find a protrusion X that is large but has small $\delta(X)$, and replace it with a smaller gadget X' that has the same behaviour. The following lemma formalizes this intuition.

▶ **Lemma 13** (⋆). *There is a constant $c_{\mathcal{F}}$ and algorithm that, given a graph G and a $2b_{\mathcal{F}}$-protrusion X in it with $\|G[X]\| > c_{\mathcal{F}}$, outputs in linear time a graph G' with $\mathsf{OPT}(G) = \mathsf{OPT}(G')$ and $\|G'\| < \|G\|$. Moreover, there is a linear-time algorithm working as follows: given a subset F' of edges of G' such that $G' - F'$ is $\mathcal{F}$-free, the algorithm computes a subset F of edges of G such that $G - F$ is $\mathcal{F}$-free and $|F| \leq |F'|$.*

The proof of Lemma 13 follows closely the strategy used by Fomin et al. [14]: Every $2b_{\mathcal{F}}$-protrusion can be assigned a type, where the number of types is bounded by a function

depending on $\mathcal{F}$ only. The type of a protrusion can be computed efficiently due to protrusions having constant treewidth. Protrusions with the same type behave in the same way with respect to the problem of our interest, and hence can be replaced by one another. Therefore, we store a replacement table consisting of the smallest protrusion of each type, so that every larger protrusion can be replaced by a smaller representative stored in the table. The lifting algorithm finds, using dynamic programming, a partial solution in the large protrusion that has the same behaviour as the given partial solution in the replacement protrusion, while being not larger. Note that while a replacement table exists and can be hard-coded into the algorithm (as it depends on the fixed family $\mathcal{F}$ only), giving an explicit bound on the size of the graphs in it (and thus on $c_{\mathcal{F}}$ in Lemma 13) would require additional arguments. The details can be found in the full version of the paper.

We henceforth define a *replaceable protrusion* in G as a $2b_{\mathcal{F}}$-protrusion X with $\|G[X]\| > c_{\mathcal{F}}$, where $c_{\mathcal{F}}$ is the constant given by Lemma 13.

Finding excessive protrusions. Recall that a replaceable protrusion in a graph G is a $2b_{\mathcal{F}}$-protrusion X with $\|G[X]\| > c_{\mathcal{F}}$. To find replaceable protrusions in the input graph, we need to assume some additional connectivity constraint (which will be implied from a connected tree-cut decomposition) – this is captured by the following definition. The larger protrusion size is needed to make any connected component of the protrusion replaceable.

▶ **Definition 14.** A $2b_{\mathcal{F}}$-protrusion B in a connected graph G is called *excessive* if $\|G[B]\| > 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$ and $G - B$ has at most two connected components.

Replaceable protrusions could be found easily if we allowed a (far worse) running time of the form $\|G\|^{\mathcal{O}(b_{\mathcal{F}})}$, but this would affect the running times in both our main results. With the above definition in hand, we use the techniques of *important cuts*, introduced by Marx [27] (see also the exposition in [8, Chapter 8.2]) and of *randomized contractions* by Chitnis et al. [6] instead. These two techniques allow us to reduce excessive protrusions: we use the randomized contractions technique to find a large enough subset of a presumed excessive protrusion, after which important cuts allow us to find a boundary that makes this subset a replaceable protrusion.

▶ **Lemma 15** ($\star$). *There is an algorithm that, given a connected graph G, runs in time $\mathcal{O}(\|G\| \log \|G\| \cdot |G|^2)$ and either correctly concludes that G does not contain any excessive protrusion, or it outputs some replaceable protrusion in G.*

We remark that we only defined excessive protrusions in connected graphs. Note that if B is an excessive protrusion in a connected component H of G, it would not necessarily be an excessive protrusion in G, since $G - B$ may have more components than $H - B$ (they are however not adjacent to B). We will say that *no component of G has an excessive protrusion* if for each connected component H of G, there is no excessive protrusion in H.

By exhaustively (at most $\|G\|$ times) executing the algorithm of Lemma 15 and replacing any obtained protrusion using Lemma 13, we can get rid of all excessive protrusions. We formalize this in the following lemma, which will serve as the abstraction of protrusion replacement in the sequel.

▶ **Lemma 16** (Exhaustive Protrusion Replacement). *There is an algorithm that, given a graph G, runs in time $\mathcal{O}(\|G\|^2 \log \|G\| \cdot |G|^2)$ and computes a graph G' such that $\mathsf{OPT}(G) = \mathsf{OPT}(G')$, $\|G'\| \leq \|G\|$, and no connected component of G' has an excessive protrusion.*

Moreover, there exists a solution-lifting algorithm that works as follows: given a subset F' of edges of G' for which $G' - F'$ is $\mathcal{F}$-free, the algorithm runs in time $\mathcal{O}(\|G\|^2)$ and outputs a subset F of edges of G such that $|F| \leq |F'|$ and $G - F$ is $\mathcal{F}$-free.

4 Constant-factor approximation

It would be ideal if just applying the Exhaustive Protrusion Replacement (Lemma 16) reduced the size of the graph to linear in OPT. Then, we would already have a linear kernel, and taking all its edges would yield a constant-factor approximation. Unfortunately, there are graphs with no excessive protrusions, where the size is not bounded linearly in OPT. To see this, observe that a large group of parallel edges is not a protrusion, so our current reduction rules will not reduce their multiplicity, even if they amount to 99% of the graph. Hence, we need to find a way to discover and account for such groups (we remark that reducing each to $\mathcal{O}(\mathsf{OPT})$ would be relatively easy, giving a quadratic kernel). More generally, the structures that turn out to be problematic are large groups of constant-size 2-protrusions attached to the same pair of vertices; a group of parallel edges is a degenerated case of this structure. To describe the problematic structures formally, we introduce the notion of a *bouquet*.

Bouquets. Let us define the following constant $d_{\mathcal{F}} := \max\{2b_{\mathcal{F}} \cdot c_{\mathcal{F}} + 2b_{\mathcal{F}},\ 3\mathsf{MAX}_{\mathcal{F}}\} + 1$ (where $\mathsf{MAX}_{\mathcal{F}} = \max_{H \in \mathcal{F}} \|H\|$). A bouquet is a family of at least $d_{\mathcal{F}}$ isomorphic 2-protrusions, while a theta is a set of at least $d_{\mathcal{F}}$ parallel edges.

▶ **Definition 17.** Consider a graph G, a set $U \subseteq V(G)$ and a family of 2-protrusions $\{S_i\}_{i \in I}$ such that for each $i \in I$:

- $N(S_i) = U$ (implying $|U| \leq 2$);
- $G[S_i]$ is connected; and
- $G[U \cup S_i]$ is isomorphic to $G[U \cup S_j]$ for all $i, j \in I$, with an isomorphism that maps each vertex of U to itself.

We call such a family a *bouquet attached to* U if it is maximal under inclusion (i.e. there is no proper superfamily which is also a bouquet) and has at least $d_{\mathcal{F}}$ elements. The edge set of the bouquet is the set of all edges incident to some S_i.

▶ **Definition 18.** For two vertices $u, v \in V(G)$, a *theta attached to* $\{u, v\}$ is a set of edges between u and v that is maximal under inclusion and has at least $d_{\mathcal{F}}$ elements.

The constant $d_{\mathcal{F}}$ is chosen so that a protrusion containing a set to which a bouquet (or theta) is attached is large enough to be excluded as an excessive protrusion, and so that any immersion of a graph of $\mathcal{F}$ cannot simultaneously intersect all elements of a bouquet. Indeed, in any immersion of some $H \in \mathcal{F}$ in a graph G, the image of an edge of H is a path in G, which visits every vertex of the bouquet's attachment at most once, and hence intersects at most three elements of the bouquet. Thus in total, the immersion model intersects at most $3 \cdot \max_{H \in \mathcal{F}} \|H\|$ elements of the bouquet or theta, which is less than $d_{\mathcal{F}}$.

We now show that the number of edges of a graph with no excessive protrusions, no bouquets, and no thetas is linearly bounded in the optimum solution size, which formalizes the intuition that only those structures prevent the graph from being a linear kernel.

▶ **Lemma 19** ($\star$). *Let G be a connected graph without excessive protrusions, bouquets, or thetas. Then G is $\mathcal{F}$-free, or $\|G\| \leq c \cdot \mathsf{OPT}(G)$, for some constant c depending on $\mathcal{F}$ only.*

The proof of Lemma 19 goes roughly as follows. Take some optimum solution F. Then $G - F$ is $\mathcal{F}$-free, so, by Theorem 11, it has a neat tree-cut decomposition $(T, \mathcal{X})$ of width at most $b_{\mathcal{F}}$. For simplicity suppose that $G - F$ is connected, so that T is a tree (the proof in the general case is essentially the same). Let M_0 be the set of all vertices of T whose bags contain a vertex incident to an edge of F; then $|M_0| \leq 2|F|$. Compute the *lowest common ancestor closure* M of M_0: start with $M := M_0$, and iteratively add to M any lowest common

ancestor of two nodes of M that is not yet included. As in Fomin et al. [14], we have that $|M| \leq 2|M_0| \leq 4|F|$, and each component of $T - M$ is adjacent to at most two nodes of M.

Let us consider some connected component T' of $T - M$, and let $X_{T'}$ be the union of the bags at the nodes of T'. Suppose first that T' is adjacent to exactly two nodes of M; note that there are at most $|M| - 1$ such components T'. Then one can easily see that $\|G[X_{T'}]\| \leq 2b_{\mathcal{F}}c_{\mathcal{F}}$, because otherwise $X_{T'}$ would be an excessive protrusion. Here, we crucially use the first condition of the neatness of T to argue that $G - X_{T'}$ has at most two connected components. Hence, the total number of edges in graphs $G[X_{T'}]$ for such components T' is bounded by $(|M| - 1) \cdot 2b_{\mathcal{F}}c_{\mathcal{F}}$, which is linear in $|F| = \mathsf{OPT}(G)$.

We are left with considering components T' that are adjacent to exactly one node of M. We again have that $\|G[X_{T'}]\| \leq 2b_F c_{\mathcal{F}}$ for each such component T', because otherwise $X_{T'}$ would be an excessive protrusion. However, a priori we do not have any bound on the number of such components. Suppose for a moment that a large number of such components T' is adjacent to the same node $t \in M$. By Lemma 8, all but a constant number of them are connected to t via edges of the decomposition with thin adhesions. Moreover, for each of these adhesions, all the (at most 2) edges of the adhesion have both endpoints in the bag X_t. Since the size of X_t is bounded by a constant, and each $X_{T'}$ induces a graph of constant size, we can infer that there is a constant number of isomorphism types for graphs $G[X_{T'}]$, together with the choice of the attachment points in X_t. So if the number of the considered components T' was very large, then some of them would form a bouquet, a contradiction.

This shows that, in fact, the number of components T' adjacent to only one vertex of M is also bounded linearly in $|M|$, so also in $\mathsf{OPT}(G)$. The fact that G has no thetas is used to bound the number of edges contained in graphs induced by the bags of M. By combining all these bounds, we conclude the proof of Lemma 19.

Finding a constant-factor approximation piece by piece. To handle bouquets and thetas, we first show that they are disjoint, as otherwise they would constitute a large protrusion.

▶ **Lemma 20** ($\star$). *Let G be a connected graph with no excessive protrusions. Then every two bouquets and/or thetas in G have disjoint edge sets. Furthermore, if a bouquet or theta is attached to $U \subseteq V(G)$, then U is disjoint with all elements of any bouquet.*

The following lemma is the crucial step for our approximation: we find a subset of edges Δ with a guarantee that a constant fraction of Δ is used in some optimum solution.

▶ **Lemma 21** ($\star$). *Given a connected graph G with no excessive protrusion that is not $\mathcal{F}$-free, one can find in in time $\mathcal{O}(|G|^3)$ a set $\Delta \subseteq E(G)$ such that (for some c depending on $\mathcal{F}$ only):* $\mathsf{OPT}(G - \Delta) < \mathsf{OPT}(G)$ *and* $|\Delta| \leq c \cdot (\mathsf{OPT}(G) - \mathsf{OPT}(G - \Delta))$.

The set Δ given by Lemma 21 is constructed as follows. First, we locate all thetas and bouquets in G; Lemma 20 ensures that they do not overlap. Δ is defined as the set of all edges of G, with the exception that in each bouquet and in each theta we exclude from Δ the edges of all but $d_{\mathcal{F}} - 1$ elements of the bouquet/theta. We show that the subgraph given by edges of Δ satisfies the assumptions of Lemma 19, thus bounding $|\Delta|$. The bound is linear in the size of the intersection of Δ with any optimal solution, allowing to conclude the claim.

To get a constant-factor approximation algorithm, we iteratively invoke the algorithm of Lemma 21 (extended to general graphs by considering each connected component separately). More precisely, we perform iteratively the following procedure, starting with $G_1 = G$. At step i, given a graph G_i, we first run the algorithm of Lemma 16 to remove excessive protrusions from G_i and obtain a new graph G_i'. Thus, in a sense, we reduce those parts

of the graph where no more edges need to be deleted. Then, we apply the algorithm of Lemma 21 to G'_i, thus finding a set Δ_i with the following property: the deletion of Δ_i reduces OPT by some number p, while increasing the total number of edges deleted so far by at most $c \cdot p$. We proceed to the next iteration with $G_{i+1} := G'_i - \Delta_i$. Eventually, we arrive at the situation when the current graph G_i is already $\mathcal{F}$-free, in which case we stop. A constant-factor approximate solution can be then obtained by reverting the iterations: Proceeding from the last iteration to the first, we always add the extracted set Δ_i to the constructed solution, and roll-back the protrusion reductions performed by the algorithm of Lemma 16 while lifting the current solution using the solution-lifting algorithm. This concludes the proof of Theorem 2 (the formal description is in the full version of the paper).

5 Linear kernel

In the previous section we observed (Lemma 19) that the only structures in the graph that prevent it from being a linear kernel are excessive protrusions, bouquets, and thetas. Using the Exhaustive Protrusion Replacement (Lemma 16) we can get rid of excessive protrusions, but bouquets and thetas can still be present in the graph. It would be ideal if we could reduce the size of every bouquet or theta to a constant, but unfortunately we are unable to do this. Instead, bouquets and thetas will be reduced to constant size in the amortized sense.

The next lemma is the crux of our approach: provided we know a "local" solution Δ that isolates a bouquet into an $\mathcal{F}$-free part, this bouquet can be pruned proportionally to the size of Δ without changing $\mathsf{OPT}(G)$. The proof is by a simple replacement argument, and the same reasoning can also be applied to limit the sizes of thetas.

▶ **Lemma 22** (⋆). *Let $\{X_i\}_{i\in I}$ be a bouquet attached to U in G. Suppose $\Delta \subseteq E(G)$ is such that all the connected components of $G - \Delta$ that intersect $U \cup \bigcup_{i\in I} X_i$ are $\mathcal{F}$-free. Then $\mathsf{OPT}(G) = \mathsf{OPT}(G')$, where G' is obtained from G by removing vertices of all except $d_{\mathcal{F}} + |\Delta|$ elements of the bouquet.*

We are now ready to show the main part of our reasoning: given some solution F, for example the one returned by the approximation algorithm of Theorem 2, we are able to reduce the graph, provided it is not already bounded linearly in $|F|$.

▶ **Lemma 23** (⋆). *Let G be a connected graph with no excessive protrusions and let $F \subseteq E(G)$ be such that $G - F$ is $\mathcal{F}$-free. Then either $\|G\| \leq c \cdot |F|$ for some constant c depending on $\mathcal{F}$ only, or given G and F, one can compute in time $\mathcal{O}(\|G\| \cdot |G|^2)$ a subgraph G' of G such that $\mathsf{OPT}(G) = \mathsf{OPT}(G')$ and $\|G'\| < \|G\|$.*

The proof uses the following strategy based on the idea of *amortization*. Given a solution F, we use parts of F as local solutions to locally bound bouquets and thetas. More precisely, we first perform a similar structural analysis of G with F removed as in the proof of Lemma 19; see the sketch following its statement. There, we considered a neat tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ of $G - F$ of width $b_{\mathcal{F}}$. In this decomposition, we highlighted a subset $M \subseteq V(T)$, the lca-closure of those nodes of T whose bags are incident to F. We concluded that the only parts not yet bounded linearly in terms of $|F|$ were large bouquets and thetas with both attachment points contained in a bag X_t of some node $t \in M$. For such a node t, define $\Delta(t)$ as the set containing: (i) all the edges of F incident to X_t, and (ii) all the adhesions of edges of T incident to t that lead to components of $T - M$ containing other vertices of M. Then $\Delta(t)$ easily satisfies the prerequisites of Lemma 22. Hence, $d_{\mathcal{F}} + |\Delta(t)|$ can be used as a bound for reducing these bouquets and thetas. Summing these bounds through all nodes $t \in M$, we achieve an $\mathcal{O}(|F| + |M|)$ bound, thus $\mathcal{O}(|F|)$ due to $|M| \leq 4|F|$.

With Lemma 23, we can now easily conclude the proof of Theorem 3 (the formal description is in the full version of the paper). First, we get rid of all excessive protrusions in the graph (Lemma 16) and compute an approximate solution F_{apx} (Theorem 2). If $|F_{\mathrm{apx}}| > c_{\mathrm{apx}} \cdot k$, the input is a NO instance. Otherwise, we give F_{apx} to the algorithm of Lemma 23 which, provided the graph is still too large to be a kernel, computes a strictly smaller, but equivalent instance. We then recurse on this smaller instance, eventually returning a linear kernel.

Acknowledgements. The authors thank an anonymous referee for suggesting a more direct approach to finding excessive protrusions as well as Ignasi Sau, Petr Golovach, Eun Jung Kim, and Christophe Paul for preliminary discussions on the $\mathcal{F}$-Immersion Deletion problem.

References

1 Rémy Belmonte, Archontia Giannopoulou, Daniel Lokshtanov, and Dimitrios M. Thilikos. The Structure of W_4-Immersion-Free Graphs. *ArXiv e-prints 1602.02002*, February 2016.

2 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

3 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *J. ACM*, 63(5):44:1–44:69, November 2016. doi:10.1145/2973749.

4 Heather D. Booth, Rajeev Govindan, Michael A. Langston, and Siddharthan Ramachandramurthi. Fast algorithms for K_4 immersion testing. *J. Algorithms*, 30(2):344–378, 1999.

5 Dimitris Chatzidimitriou, Jean-Florent Raymond, Ignasi Sau, and Dimitrios M. Thilikos. An $O(\log \mathsf{OPT})$-approximation for covering/packing minor models of θ_r. *Algorithmica*, 2017. To appear.

6 Rajesh Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016.

7 Bruno Courcelle. The Monadic Second-Order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

8 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

9 Matt Devos, Zdeněk Dvořák, Jacob Fox, Jessica McDonald, Bojan Mohar, and Diego Scheide. A minimum degree condition forcing complete graph immersion. *Combinatorica*, 34(3):279–298, 2014.

10 Zdeněk Dvořák and Paul Wollan. A structure theorem for strong immersions. *J. Graph Theory*, 83(2):152–163, 2016. doi:10.1002/jgt.21990.

11 Zdeněk Dvořák and Liana Yepremyan. Complete graph immersions and minimum degree. *ArXiv e-prints 1512.00513*, December 2015.

12 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

13 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-Deletion: Approximation and optimal FPT algorithms. *ArXiv e-prints 1204.4230*, October 2012.

14 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-deletion: Approximation, kernelization and optimal FPT algorithms. In *Proceedings of FOCS 2012*, pages 470–479. IEEE Computer Society, 2012.

15 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *Proceedings of SODA 2010*, pages 503–510. SIAM, 2010.

16 Robert Ganian, Eun Jung Kim, and Stefan Szeider. Algorithmic applications of tree-cut width. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Proceedings of MFCS 2015*, volume 9235 of *Lecture Notes in Computer Science*, pages 348–360. Springer, 2015.

17 Archontia C. Giannopoulou, Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. Uniform kernelization complexity of hitting forbidden minors. *ACM Trans. Algorithms*, 13(3):35:1–35:35, March 2017. doi:10.1145/3029051.

18 Archontia C. Giannopoulou, O-joung Kwon, Jean-Florent Raymond, and Dimitrios M. Thilikos. Packing and covering immersion models of planar subcubic graphs. In *Proceedings of WG 2016*, pages 74–84. Springer, 2016. Preprint: *ArXiv e-prints 1602.04042*.

19 Archontia C. Giannopoulou, Michał Pilipczuk, Dimitrios M. Thilikos, Jean-Florent Raymond, and Marcin Wrochna. Linear kernels for edge deletion problems to immersion-closed graph classes. *ArXiv e-prints 1609.07780*, September 2016. URL: https://arxiv.org/abs/1609.07780.

20 Archontia C. Giannopoulou, Iosif Salem, and Dimitris Zoros. Effective computation of immersion obstructions for unions of graph classes. *J. Comput. Syst. Sci.*, 80(1):207–216, 2014.

21 Rajeev Govindan and Siddharthan Ramachandramurthi. A weak immersion relation on graphs and its applications. *Disc. Math.*, 230(1–3):189–206, 2001.

22 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of STOC 2011*, pages 479–488. ACM, 2011.

23 Ananth V. Iyer, H. Donald Ratliff, and Gopalakrishnan Vijayan. On an edge ranking problem of trees and graphs. *Discrete Appl. Math.*, 30(1):43–52, 1991.

24 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. In *Proceedings of ICALP 2013*, volume 7965 of *Lecture Notes in Computer Science*, pages 613–624. Springer, 2013.

25 Eun Jung Kim, Sang-il Oum, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. An FPT 2-approximation for tree-cut decomposition. *Algorithmica*, pages 1–20, 2016. doi:10.1007/s00453-016-0245-5.

26 Tak Wah Lam and Fung Ling Yue. Edge ranking of graphs is hard. *Discrete Appl. Math.*, 85(1):71–86, 1998.

27 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

28 Dániel Marx and Paul Wollan. Immersions in highly edge connected graphs. *SIAM J. Discrete Math.*, 28(1):503–520, 2014.

29 Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

30 N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

31 Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.

32 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.

33 Neil Robertson and Paul D. Seymour. Graph minors. XXIII. Nash-Williams' immersion conjecture. *J. Comb. Theory, Ser. B*, 100(2):181–205, 2010.

34 Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

35 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *J. Algorithms*, 56(1):1–24, 2005.

36 Paul Wollan. The structure of graphs not admitting a fixed immersion. *J. Comb. Theory, Ser. B*, 110:47–66, 2015.

k-Distinct In- and Out-Branchings in Digraphs*†

Gregory Gutin[1], Felix Reidl[2], and Magnus Wahlström[3]

1 **Royal Holloway, University of London, London, UK**
g.gutin@rhul.ac.uk
2 **North Carolina State University, Raleigh, NC, USA**
felix.reidl@gmail.com
3 **Royal Holloway, University of London, London, UK**
Magnus.Wahlstrom@rhul.ac.uk

Abstract

An out-branching and an in-branching of a digraph D are called k-distinct if each of them has k arcs absent in the other. Bang-Jensen, Saurabh and Simonsen (2016) proved that the problem of deciding whether a strongly connected digraph D has k-distinct out-branching and in- branching is fixed-parameter tractable (FPT) when parameterized by k. They asked whether the problem remains FPT when extended to arbitrary digraphs. Bang-Jensen and Yeo (2008) asked whether the same problem is FPT when the out-branching and in-branching have the same root.

By linking the two problems with the problem of whether a digraph has an out-branching with at least k leaves (a leaf is a vertex of out-degree zero), we first solve the problem of Bang-Jensen and Yeo (2008). We then develop a new digraph decomposition called the rooted cut decomposition and using it we prove that the problem of Bang-Jensen et al. (2016) is FPT for all digraphs. We believe that the *rooted cut decomposition* will be useful for obtaining other results on digraphs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Digraphs, Branchings, Decompositions, FPT algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.58

1 Introduction

While both undirected and directed graphs are important in many applications, there are significantly more algorithmic and structural results for undirected graphs than for directed ones. The main reason is likely to be the fact that most problems on digraphs are harder than those on undirected graphs. The situation has begun to change: recently there appeared a number of important structural results on digraphs, see e.g. [16, 17, 18]. However, the progress was less pronounced with algorithmic results on digraphs, in particular, in the area of parameterized algorithms.

In this paper, we introduce a new decomposition for digraphs and show its usefulness by solving an open parameterized problem on digraphs by Bang-Jensen, Saurabh and Simonsen [6]. We believe that our decomposition will prove to be helpful for obtaining further algorithmic and structural results on digraphs.

A digraph T is an *out-tree* (an *in-tree*) if T is an oriented tree with just one vertex s of in-degree zero (out-degree zero). The vertex s is the *root* of T. A vertex v of an out-tree

* A full version of the paper is available at https://arxiv.org/abs/1612.03607.
† Gutin's research was partially supported by Royal Society Wolfson Research Merit Award.


(in-tree) is called a *leaf* if it has out-degree (in-degree) zero. If an out-tree (in-tree) T is a spanning subgraph of a digraph D, then T is an *out-branching* (an *in-branching*) of D. It is well-known that a digraph D contains an out-branching (in-branching) if and only if D has only one strongly connected component with no incoming (no outgoing) arc [3].

A well-known result in digraph algorithms, due to Edmonds, states that given a digraph D and a positive integer ℓ, we can decide whether D has ℓ arc-disjoint out-branchings in polynomial time [15]. The same result holds for ℓ arc-disjoint in-branchings. Inspired by this fact, it is natural to ask for a "mixture" of out- and in-branchings: given a digraph D and a pair u, v of (not necessarily distinct) vertices, decide whether D has an arc-disjoint out-branching T_u^+ rooted at u and an in-branching T_v^- rooted at v. We will call this problem Arc- Disjoint Branchings.

Thomassen proved (see [2]) that the problem is NP-complete and remains NP-complete if we add the condition that $u = v$. The same result still holds for digraphs in which the out-degree and in-degree of every vertex equals two [7]. The problem is polynomial-time solvable for tournaments [2] and for acyclic digraphs [8, 10]. The single-root special case (i.e., when $u = v$) of the problem is polynomial time solvable for quasi-transitive digraphs[1] [4] and for locally semicomplete digraphs[2] [5].

An out-branching T^+ and an in-branching T^- are called *k-distinct* if $|A(T^+) \setminus A(T^-)| \geq k$. Bang-Jensen, Saurabh and Simonsen [6] considered the following parameterization of Arc-Disjoint Branchings.

k-Distinct Branchings parametrised by k

Input:	A digraph D, an integer k.
Problem:	Are there k-distinct out-branching T^+ and in-branching T^-?

They proved that k-Distinct Branchings is fixed-parameter tractable (FPT)[3] when D is strongly connected and conjectured that the same holds when D is an arbitrary digraph. Earlier, Bang-Jensen and Yeo [9] considered the version of k-Distinct Branchings where T^+ and T^- must have the same root and asked whether this version of k-Distinct Branchings, which we call Single-Root k-Distinct Branchings, is FPT.

The first key idea of this paper is to relate k-Distinct Branchings to the problem of deciding whether a digraph has an out-branching with at least k leaves via a simple lemma (see Lemma 4). The lemma and the following two results on out-branchings with at least k leaves allow us to solve the problem of Bang-Jensen and Yeo [9] and to provide a shorter proof for the above-mentioned result of Bang-Jensen, Saurabh and Simonsen [6] (see Theorem 6).

▶ **Theorem 1** ([1]). *Let D be a strongly connected digraph. If D has no out-branching with at least k leaves, then the (undirected) pathwidth of D is bounded by $O(k \log k)$.*

▶ **Theorem 2** ([12, 19]). *We can decide whether a digraph D has an out-branching with at least k leaves in time $O^*(4^k)$.*

---

[1] A digraph $D = (V, A)$ is quasi-transitive if for every $xy, yz \in A$ there is at least one arc between x and z, i.e. either $xz \in A$ or $zx \in A$ or both.

[2] A digraph $D = (V, A)$ is locally semicomplete if for every $xy, xz \in A$ there is at least one arc between y and z and for every $yx, zx \in A$ there is at least one arc between y and z. Tournaments and directed cycles are locally semicomplete digraphs.

[3] Fixed-parameter tractability of k-Distinct Branchings means that the problem can be solved by an algorithm of runtime $O^*(f(k))$, where O^* omits not only constant factors, but also polynomial ones, and f is an arbitrary computable function. The books [11, 13] are excellent recent introductions to parameterized algorithms and complexity.

The general case of k-Distinct Branchings seems to be much more complicated. We first introduce a version of k-Distinct Branchings called k-Rooted Distinct Branchings, where the roots s and t of T^+ and T^- are fixed, and add the arc ts to D (provided the arc is not in D) to make D strongly connected. This introduces a complication: we may end up in a situation where D has an out-branching with many leaves, and thereby potentially unbounded pathwidth, but the root of the out-branching is not s. To deal with this situation, our goal will be to *reconfigure* the out-branching into an out-branching rooted at s. In order to reason about this process, we develop a new digraph decomposition we call the *rooted cut decomposition*. The cut decomposition of a digraph D rooted at a given vertex r consists of a tree $\hat{T}$ rooted at r whose nodes are some vertices of D and subsets of vertices of D called *diblocks* associated with the nodes of $\hat{T}$.

Out strategy is now as follows. If $\hat{T}$ is *shallow* (i.e., it has bounded height), then any out-branching with sufficiently many leaves can be turned into an out-branching rooted at s without losing too many of the leaves. On the other hand, if $\hat{T}$ contains a path from the root of $\hat{T}$ with sufficiently many non-degenerate diblocks (diblocks with at least three vertices), then we are able to show immediately that the instance is positive. The remaining and most difficult issue is to deal with digraphs with decomposition trees that contain long paths of diblocks with only two vertices, called *degenerate* diblocks. In this case, we employ two reduction rules which lead to decomposition trees of bounded height.

The paper is organized as follows. In the next section, we provide some terminology and notation on digraphs used in this paper. In Section 3, we prove Theorem 6. Section 4 is devoted to proving that Rooted k-Distinct Branchings is FPT for all digraphs using cut decomposition and Theorems 1 and 2. We conclude the paper in Section 5, where some open parameterized problems on digraphs are mentioned. Due to space constraints, we only provide a short version of Section 4.3 in the main text. A complete version is available online[4].

2 Terminology and Notation

Let us recall some basic terminology of digraph theory, see [3]. A digraph D is *strongly connected* (*connected*) if there is a directed (oriented) path from x to y for every ordered pair x, y of vertices of D. Equivalently, D is connected if the underlying graph of D is connected. A vertex v is a *source* (*sink*) is its in-degree (out-degree) is equal to zero. It is well-known that every acyclic digraph has a source and a sink [3].

In this paper, we exclusively work with digraphs, therefore we assume all our graphs, paths, and trees to be directed unless otherwise noted. For a path $P = x_1x_2 \dots x_k$ of length $k-1$ we will employ the following notation for subpaths of P: $P[x_i, x_j] := x_i \dots x_j$ for $1 \leq i \leq j \leq k$ is the *infix* of P from x_i to x_j. For paths $P_1 := x_1 \dots x_k v$ and $P_2 := vy_1 \dots y_\ell$ we denote by $P_1P_2 := x_1 \dots x_k v y_1 \dots y_\ell$ their *concatenation*. For rooted trees T and some vertex $x \in T$, T_x stands for the subtree of T rooted at x (see Figure 1).

We will frequently partition the nodes of a tree around a path in the following sense (*cf.* Figure 1): Let T be a tree rooted at r and $P = x_1 \dots x_\ell$ a path from $r = x_1$ to some node $x_\ell \in T$. The *fins* of P are the sets $\{F_{x_i}\}_{x_i \in P}$ defined as $F_{x_i} := V(T_{x_i}) \setminus V(T_{x_{i+1}})$ for $i < \ell$ and $F_{x_\ell} := V(T_{x_\ell})$.

▶ **Definition 3** (Bi-reachable Set)**.** A set B in a digraph D is *bi-reachable* from a vertex r if for all $v \in B$ there exist two internally vertex-disjoint paths from r to v.

[4] https://arxiv.org/abs/1612.03607

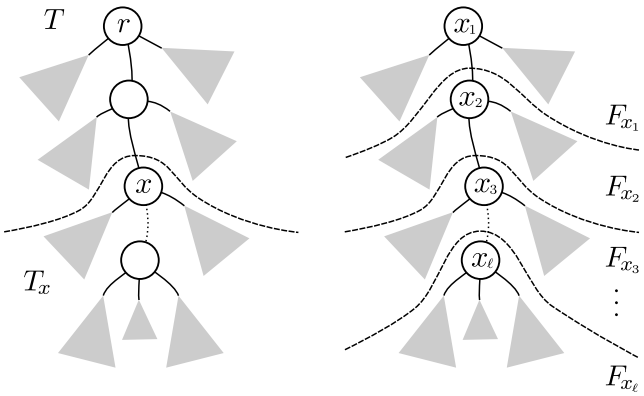


Figure 1 Subtree notation T_x for $x \in T$ (left) and the fins $F_{x_1}, \ldots, F_{x_\ell}$ for a path $x_1 \ldots x_\ell$ in T (right).

Given a digraph D and a vertex r, we can compute the set of vertices that are bi-reachable from r in polynomial time using network flows.

3 Strongly Connected Digraphs

Let us prove a simple fact on a link between out/in-branchings with many leaves and k-Distinct Branchings, which together with a structural result of Alon *et al.* [1] and an algorithmic result for the maximum leaf out-branching problem [12, 19] gives a short proof that both versions of k-Distinct Branchings are FPT for strongly connected digraphs.

▶ **Lemma 4.** *Let D be a digraph containing an out-branching and an in-branching. If D contains an out-branching (in-branching) T with at least $k+1$ leaves, then every in-branching (out-branching) T' of D is k-distinct from T.*

Proof. We will consider only the case when T is an out-branching since the other case can be treated similarly. Let T' be an in-branching of D and let L be the set of all leaves of T apart from the one which is the root of T'. Observe that all vertices of L have outgoing arcs in T' and since in T the incoming arcs of L are the only arcs incident to L in T, the sets of the outgoing arcs in T' and incoming arcs in T do not intersect. ◀

We will use the following standard dynamic programming result (see, e.g., [6]).

▶ **Lemma 5.** *Let H be a digraph of (undirected) treewidth τ. Then k-Distinct Branchings and Single-Root k-Distinct Branchings on H can be solved in time $O^*(2^{O(\tau \log \tau)})$.*

Note that if a digraph D is a positive instance of Single-Root k-Distinct Branchings then D must be strongly connected as an out-branching and an in-branching rooted at the same vertex form a strongly connected subgraph of D.

▶ **Theorem 6.** *k-Distinct Branchings and Single-Root k-Distinct Branchings on strongly connected digraphs can be solved in time $O^*(2^{O(k \log^2 k)})$.*

Proof. The proof is essentially the same for both problems and we will give it for Single-Root k-Distinct Branchings. Let D be an input strongly connected digraph. By Theorem 2 using an $O^*(4^k)$-time algorithm we can find an out-branching T^+ with at least $k+1$ leaves, or decide that D has no such out-branching. If T^+ is found, the instance of

Single-Root k-Distinct Branchings is positive by Lemma 4 as any in-branching T^- of D is k-distinct from T^+. In particular, we may assume that T^- has the same root as T^+ (a strongly connected digraph has an in-branching rooted at any vertex). Now suppose that T^+ does not exist. Then, by Theorem 1 the (undirected) pathwidth of D is bounded by $O(k \log k)$. Thus, by Lemma 5 the instance can be solved in time $O^*(2^{O(k \log^2 k)})$. ◀

4 The k-Distinct Branchings Problem

In this section, we fix a digraph D with terminals s, t and simply talk about *rooted out-branchings (in-branchings)* whose root we implicitly assume to be s (t). Similarly, unless otherwise noted, a *rooted out-tree (in-tree)* is understood to be rooted at s (t).

The problem k-Distinct Branchings in which T^+ and T^- must be rooted at s and t, respectively, will be called the Rooted k-Distinct Branchings problem. Clearly, to show that both versions of k-Distinct Branchings are FPT it is sufficient to prove the following:

▶ **Theorem 7.** Rooted k-Distinct Branchings *is FPT for arbitrary digraphs.*

In the rest of this section, (D, s, t) will stand for an instance of Rooted k-Distinct Branchings (in particular, D is an input digraph of the problem) and H for an arbitrary digraph. As noted in the previous section, the case in which $s = t$ implies strong connectivity and is therefore already solved. Consequently, we will assume that $s \neq t$ in the following. Let us start by observing what further restrictions on D can be imposed by polynomial-time preprocessing.

4.1 Preprocessing

Let (D, s, t) be an instance of Rooted k-Distinct Branchings with $s \neq t$. Recall that D contains an out-branching (in-branching) if and only if D has only one strongly connected component with no incoming (no outgoing) arc. As a first preprocessing step, we can decide in polynomial time whether D has a rooted out-branching and a rooted in-branching. If not, we reject the instance. Note that this in particular means that in a non-rejected instance, every vertex in D is reachable from s and t is reachable from every vertex.

Next, we test for every arc $a \in D$ whether there exists at least one rooted in- or out-branching that uses a as follows: since a maximal-weight out- or in-branching for an arc-weighted digraph can be computed in polynomial time [14], we can force the arc a to be contained in a solution by assigning it a weight of 2 and every other arc weight 1. If we verify that a indeed neither appears in any rooted out-branching or in-branching, we remove a from D and obtain an equivalent instance of Rooted k-Distinct Branchings.

After this polynomial-time preprocessing, our instance has the following properties: there exists a rooted out-branching, there exists a rooted in-branching, and every arc of D appears in some rooted in- or out-branching. We call such a digraph with a pair s, t *reduced*.

Lastly, by the following lemma we may assume that our instance is strongly connected by incurring a factor of two in the application of Lemma 4.

▶ **Lemma 8.** *Let (D, s, t) be reduced and let D' be the digraph obtained from D by adding the arc ts to it unless ts is already in D in which case $D' = D$. Then (D', s, t) is a positive instance of* Rooted k-Distinct Branchings *if and only if so is (D, s, t). Furthermore, if D' contains an out-tree (in-tree) with at least ℓ leaves, then D contains an out-tree (in-tree) with at least $\ell/2$ leaves.*

Proof. We may assume that $D' \neq D$. For the first claim, simply note that any rooted out-branching of D' cannot use the arc ts and the same holds for rooted in-branchings. For the second claim, assume T is an out-tree in D with ℓ leaves. Assume $ts \in T$ (otherwise the claim follows trivially). Let T_1 and T_2 be the two out-trees obtained by deleting the arc ts from T. Both are out-trees in D and one of them contains at least $\ell/2$ leaves, as claimed. ◀

The first claim of Lemma 8 shows that we may assume that D is strongly connected in Rooted k-Distinct Branchings. This implies the following simple claim required for further references.

▶ **Lemma 9.** *Let D be an input digraph of* Rooted k-Distinct Branchings. *Then every rooted out-tree with q leaves can be extended into a rooted out-branching with at least q leaves.*

In summary, we enforce the following properties for (D, s, t) by polynomial- time preprocessing:

1. Every arc of D is contained in at least one rooted in-branching or rooted out-branching,
2. D is strongly connected.

4.2 Decomposition and Reconfiguration

We work towards the following win-win scenario: either we find an out-tree with $\Theta(k)$ leaves that can be turned into a rooted out-tree with at least $k+1$ leaves, or we conclude that every out-tree in D has less than $\Theta(k)$ leaves. We refer to the process of turning an out-tree into a rooted out-tree as a *reconfiguration*. In the process we will develop a new digraph decomposition, the *rooted cut-decomposition*, which will aid us in reasoning about reconfiguration steps and ultimately lead us to a solution for the problem.

To make the notion of a bi-reachable set easier to use, the decomposition will employ a slightly broader notion as follows.

▶ **Definition 10.** Let H be a digraph with at least two vertices, and let $r \in V(H)$ such that every vertex of H is reachable from r. Let $B \subseteq V(H)$ be the set of all vertices that are bi-reachable from r. The *directed block (diblock) B_r of r in H* is the set $B \cup N^+[r]$, i.e., the bi-reachable vertices together with all out-neighbors of r and r itself.

Note that according to the above definition a diblock must have at least two vertices.

The following statement provides us with an easy case in which a reconfiguration is successful, that is, we can turn an arbitrary out-tree into a rooted out-tree without losing too many leaves. Later, the obstructions to this case will be turned into building blocks of the decomposition.

▶ **Lemma 11.** *Let $B_s \subseteq V(D)$ be the diblock of s and let T be an out-tree of D whose root r lies in B_s with ℓ leaves. Then there exists a rooted out-tree with at least $(\ell-1)/2$ leaves.*

Proof. We may assume that $r \neq s$. In case T contains s as a leaf, we remove s from T for the remaining argument and hence will argue about the $\ell - 1$ remaining leaves.

If r is bi-reachable from s, consider two internally vertex-disjoint paths P, Q from s to r. One of the two paths necessarily avoids half of the $\ell - 1$ leaves of T; let without loss of generality this path be P. Let further L be the set of those leaves of T that do *not* lie on P. If $r \in N^+(s)$, let $P = sr$.

We construct the required out-tree T' as follows: first, add all arcs and vertices of P to T'. Now for every leaf $v \in L$, let P_v be the unique path from r to v in T and let P'_v be the segment of P_v from the last vertex x of P_v contained in T. Add all arcs and vertices

of P'_v to T'. Observe that $x \neq v$ as v cannot be in T' already. Since P_v and thus P'_v contains no leaf of L other than v, in the end of the process, all vertices of L are leaves of T'. Since $|L| \geq (\ell - 1)/2$, the claim follows. ◀

Note that the definition of diblocks can be understood in terms of network flows. Let $v \neq r$. Consider the vertex-capacitated version of H where r and v both have capacity 2, and every other vertex has capacity 1, for some $v \in V(H) \setminus \{r\}$. Then v is contained in the diblock of r in H if and only if the max-flow from r to v equals 2 (note that if $v \in N^+(r)$, then such a flow exists trivially since arcs have infinite capacity). Dually, by Menger's theorem, v is *not* contained in the diblock if and only if there is a vertex $u \notin \{r, v\}$ such that all r-v paths P intersect u. This has the following simple consequence regarding connectivity inside a diblock.

▶ **Lemma 12.** *Fix $r \in V(H)$ and let $B_r \subseteq V(H)$ be the diblock of r in H. Then for every pair of distinct vertices $x, y \in B_r$, there exist an r-x-path P_x and an r-y-path P_y that intersect only in r.*

Proof. If $r \in \{x, y\}$, then clearly the claim holds since every vertex in B_r is reachable from r. Otherwise, add a new vertex z with arcs xz and yz, and note that the lemma holds if and only if z is bi-reachable from r. If this is not true, then by Menger's theorem there is a vertex $v \in B_r$, $v \neq r$, such that all paths from r to z, and hence to x and y, go through v. But as noted above, there is no cut-vertex $v \notin \{x, r\}$ for r-x paths, and no cut-vertex $v \notin \{y, r\}$ for r-y paths. We conclude that z is bi-reachable from r, hence the lemma holds. ◀

Next, we will use Lemma 12 to show that given a vertex r, the set of vertices not in the diblock B_r of r in H partitions cleanly around B_r.

▶ **Lemma 13.** *Let $r \in V(H)$ be given, such that every vertex of H is reachable from r. Let $B_r \subset V(H)$ be the diblock of r in H. Then $V(H) \setminus B_r$ partitions according to cut vertices in B_r, in the following sense: For every $v \in V(H) \setminus B_r$, there is a unique vertex $x \in B_r \setminus \{r\}$ such that every path from r to v intersects B_r for the last time in x. Furthermore, this partition can be computed in polynomial time.*

Proof. Assume towards a contradiction that for $v \in V(H) \setminus B_r$ there exist two r-v-paths P_1, P_2 that intersect B_r for the last time in distinct vertices x_1, x_2, respectively. We first observe that $r \notin \{x_1, x_2\}$, since the second vertices of P_1 and P_2 are contained in B_r by definition. By Lemma 12, we may assume that $P_1[r, x_1] \cap P_2[r, x_2] = \{r\}$. But then P_1 and P_2 intersect for the first time outside of B_r in some vertex v' (potentially in $v' = v$). This vertex is, however, bi-reachable from r, contradicting our construction of B_r. Hence there is a vertex $x \in B_r$ such that every path from r to v intersects B_r for the last time in x, with $x \neq r$, and clearly this vertex is unique. Finally, the set B_r can be computed in polynomial time, and given B_r it is easy to compute for each $x \in B_r$ the set of all vertices $v \in V(H)$ (if any) for which x is a cut vertex. ◀

We refer to the vertices $x \in B_r$ that are cut vertices in the above partition as the *bottlenecks of B_r*. Note that r itself is not considered a bottleneck in B_r. Using these notions, we can now define a *cut decomposition* of a digraph H.

▶ **Definition 14** (Rooted cut decomposition and its tree)**.** Let H be a digraph and r a vertex such that every vertex in H is reachable from r. The *(r-rooted) cut decomposition* of H is a pair $(\hat{T}, \mathcal{B})$ where $\hat{T}$ is a rooted tree with $V(\hat{T}) \subseteq V(H)$ and $\mathcal{B} = \{B_x\}_{x \in \hat{T}}$, $B_x \subseteq V(H)$ for each $x \in \hat{T}$, is a collection of diblocks associated with the nodes of $\hat{T}$, defined and computed recursively as follows.

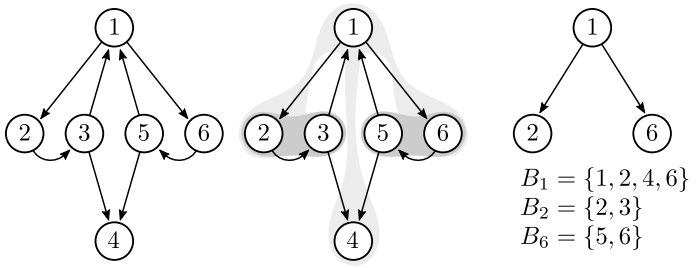


Figure 2 An example of a rooted cut decomposition.

1. Let B_r be the diblock of r in H, and let $L \subseteq B_r \setminus \{r\}$ be the set of bottlenecks in B_r. Let $\{X_x\}_{x \in L}$ be the corresponding partition of $V(H) \setminus B_r$.
2. For every $x \in L$, let $(\hat{T}_x, \mathcal{B}_x)$ be the x-rooted cut decomposition of $H[X_x \cup \{x\}]$.
3. $\hat{T}$ is the tree with root node r, where L is the set of children of r, and for every $x \in L$ the subtree of $\hat{T}$ rooted at x is $\hat{T}_x$.
4. $\mathcal{B} = \{B_r\} \cup \bigcup_{x \in L} \mathcal{B}_x$.

Furthermore, for every node $x \in \hat{T}$, we define $B^*_x = \bigcup_{y \in \hat{T}_x} B_y$ as the set of all vertices contained in diblocks associated with nodes of the subtree $\hat{T}_x$.

Figure 2 provides an illustration to Definition 14.

▶ **Lemma 15.** *Let a digraph H and a root $r \in V(H)$ be given, such that every vertex of H is reachable from r. Then the r-rooted cut decomposition $(\hat{T}, \{B_x\}_{x \in \hat{T}})$ of H is well-defined and can be computed in polynomial time. Furthermore, the diblocks cover $V(H)$, i.e., $\bigcup_{x \in \hat{T}} B_x = V(H)$, and for every node $x \in \hat{T}$, every vertex of B^*_x is reachable from x in $H[B^*_x]$.*

Proof. By Lemma 13, the root diblock B_r as well as the set $L \subseteq B_r$ of bottlenecks and the partition $\{X_x\}_{x \in L}$ are well-defined and can be computed in polynomial time. Also note that for each $x \in L$, $r \notin X_x \cup \{x\}$, and every vertex of $H_x := H[X_x \cup \{x\}]$ is reachable from x in H_x by the definition of the partition. Hence the collection of recursive calls made in the construction is well-defined, and every digraph H_x used in a recursive call is smaller than H, hence the process terminates. Finally, for any two distinct bottlenecks $x, y \in L$ we have $V(H_x) \cap V(H_y) = \emptyset$. Thereby, distinct nodes of $\hat{T}$ are associated with distinct vertices of H, $|\hat{T}| \leq |V(H)|$, and the map $x \mapsto B_x$ is well-defined. It is also clear that the whole process takes polynomial time. ◀

We collect some basic facts about cut decompositions.

▶ **Lemma 16.** *Let a digraph H and a vertex $r \in V(H)$ be given, and let $(\hat{T}, \{B_x\}_{x \in \hat{T}})$ be the r-rooted cut decomposition of H. Then the following hold.*

1. *The sets $\{B_x \setminus \{x\}\}_{x \in \hat{T}}$ are all non-empty and partition $V(H) \setminus \{r\}$.*
2. *For distinct nodes $x, y \in \hat{T}$, if x is the parent of y in $\hat{T}$ then $B_x \cap B_y = \{y\}$; in every other situation, $B_x \cap B_y = \emptyset$.*
3. *For every node $x \in \hat{T}$, the following hold:*
   - **(a)** *If y is a child of x in $\hat{T}$, then any arc leading into the set B^*_y from $V(H) \setminus B^*_y$ will have the form uy where $u \in B_x$.*
   - **(b)** *If y, y' are distinct children of x in $\hat{T}$, then there is no arc between B^*_y and $B^*_{y'}$.*

In particular, every arc of H is either contained in a subgraph of H induced by a diblock B_x, or it is a back arc *going from a diblock B_y to a diblock B_x, where x is an ancestor of y in $\hat{T}$.*

Proof. For the first claim, the sets $B_x \setminus \{x\}$ are non-empty by definition; we show the partitioning claim. By Lemma 13, for every $v \in V(H) \setminus \{r\}$ either $v \in B_r \setminus \{r\}$ or there is exactly one bottleneck $x \in B_r$ such that $v \in X_x$ in the construction of the decomposition. Also note that in the latter case, $v \neq x$ since $x \in B_r$. Applying the argument recursively and using that the diblocks cover $V(H)$, by Lemma 15, we complete the proof of the partitioning claim.

For the second claim, the partitioning claim implies that if $v \in B_x \cap B_y$ for distinct nodes $x, y \in \hat{T}$, then either $v = x$ or $v = y$, i.e., v must be a bottleneck. This is only possible in the situation described.

For Claim 3(b), first consider the diblock B_r and the partition $\{X_x\}$ given by Lemma 13. We show that for any two distinct sets X_x, X_y of the partition, there is no arc between X_x and X_y. Suppose for a contradiction that there is such an arc uv, $u \in X_x$, $v \in X_y$. By Lemma 12, there are paths P_x and P_y in B_r that intersect only in r, and by Lemma 15, there are paths P_u from x to u in X_x and P_v from y to v in X_y. But then the paths $P_x P_u uv$ and $P_y P_v$ form two r-v paths that are internally vertex-disjoint, showing that $v \in B_r$, contrary to our assumptions. Since the decomposition is computed recursively, this also holds in every internal node of $\hat{T}$.

For Claim 3(a), let uv be an arc such that $u \notin B_y^*$ and $v \in B_y^*$. Moreover, let $u \in B_{x'}$ and $v \in B_{y'}$. By construction of cut decomposition, there is a path $\hat{P}$ from x' to y' in $\hat{T}$ containing nodes x and y. Let x'' be the second node in $\hat{P}$ (just after x'). Thus, there is a path P from x'' to v in H containing the vertices of $\hat{P}$ apart from x'.

Assume that $u \neq x''$. Then by Lemma 12, there is an x'-u-path P' and an x'-x''-path P'' of H which intersect only at x'. Then $x'P'uv$ and $P''P$ are internally vertex-disjoint paths from x' to v. This means that v must be in $B_{x'}$, a contradiction unless $x' = x$, $u \in B_x$ and $v = y$. If $u = x''$, then P and uv are internally vertex-disjoint paths from u to v. This means that v must be in $B_{x''}$, a contradiction unless $x' = x$ and $v = y$. ◀

As we saw, for every diblock B_y, $y \in \hat{T}$, any path "into" the diblock must go via the bottleneck vertex y. By induction, for any $v \in B_y$, every node of $\hat{T}$ from r to y represents a bottleneck vertex that is unavoidable for paths from r to v. More formally, the following holds in cut decompositions:

▶ **Lemma 17.** *Let $(\hat{T}, \{B_x\}_{x \in \hat{T}})$ be the cut decomposition of H rooted at r. Fix a diblock B_x for $x \in \hat{T}$. Consider a path P in H from r to $v \in B_x$ and let $x_1 \dots x_\ell$ be the sequence of bottleneck vertices that P encounters. Then $\hat{P} = x_0 x_1 \dots x_\ell$ with $x_0 = r$ is the path from r to x in $\hat{T}$.*

Proof. We prove the claim by induction over the depth d of x in $\hat{T}$. If $r = x$ then any path from r to $v \in B_r$ contains r itself and hence the base case for $d = 0$ holds.

Consider a diblock B_x, $x \in \hat{T}$ where x has distance d to r in $\hat{T}$ and let y be the parent of x in $\hat{T}$. We assume the induction hypothesis holds for diblocks at depth $d - 1$, hence it holds for B_y in particular. Because $x \in B_y$, this implies that every path from r to x will contain all ancestors of x in $\hat{T}$. Since by construction every path from r to a vertex $v \in B_x$ needs to pass through x, the inductive step holds. This proves the claim. ◀

As an immediate consequence, we can identify arcs in cut decompositions that cannot participate in any rooted out-branching.

▶ **Corollary 18.** *Let $(\hat{T}, \{B_x\}_{x \in \hat{T}})$ be the cut decomposition of rooted at r and let $R := \{uv \in A(H) \mid u \in B_x \text{ and } x \in \hat{T}_v\}$ be all the arcs that originate in a diblock B_x and end in an ancestor v of x on $\hat{T}$. Then for every out-tree T rooted at r we have $A(T) \cap R = \emptyset$.*

Proof. Fix a bottleneck vertex $v \in \hat{T}$ of the decomposition and let the arc uv be in an out-tree T rooted at r. There must exist a path P_{su} from s to u that is part of T. By Lemma 17, this path will contain the vertex v. But then v is an ancestor of u in T and therefore the arc uv cannot be part of T, which is a contradiction. ◀

The decomposition actually restricts paths even further: a path that starts at the root and visits two bottleneck vertices x, y (in this order) cannot intersect any vertex of B^*_y before visiting y and cannot return to any set B^*_z, $z \in \hat{T}$, after having left it.

▶ **Lemma 19.** *Let $(\hat{T}, \{B_x\}_{x\in\hat{T}})$ be the cut decomposition of H rooted at r. Fix a diblock B_x for $x \in \hat{T}$. Consider a path P from r to $v \in B_x$ and let $\hat{P} = x_0 \dots x_\ell$ be the path from $r = x_0$ to $x = x_\ell$ in $\hat{T}$. Let further $F_0, \dots, F_\ell$ be the fins of $\hat{P}$ in $\hat{T}$. Then the subpath $P[x_i, x_{i+1}] \setminus \{x_{i+1}\}$ is contained in the union of diblocks of F_i for $0 \le i < \ell$.*

Proof. By Lemma 17 we know that the nodes of $\hat{P}$ appear in P in the correct order, hence the subpath $P[x_i, x_{i+1}]$ is well-defined. Let us first show that the subpath $P[x_i, x_{i+1}] \setminus \{x_{i+1}\}$ cannot intersect any diblock associated with $\hat{T}_{x_{i+1}}$. By Lemma 16, the only arcs from B_{x_i} into diblocks below x_{i+1} connect to the bottleneck x_{i+1} itself. Since x_{i+1} is already the endpoint of $P[x_i, x_{i+1}]$, this subpath cannot intersect the diblocks of $\hat{T}_{x_{i+1}}$. This already proves the claim for x_0; it remains to show that it does not intersect diblocks of $V(\hat{T}) \setminus V(\hat{T}_{x_i})$ for $i \ge 1$. The reason is similar: since the bottleneck x_i is already part of $P[x_i, x_{i+1}]$, this subpath could not revisit B_{x_i} if it enters any diblock B_y for a proper ancestor y of x_i in $\hat{T}$. We conclude that therefore it must be, with the exception of the vertex x_{i+1}, inside the diblocks of the fin F_i. ◀

▶ **Corollary 20.** *For every vertex $u \in V(H)$ and every set $X \subseteq V(H) \setminus (V(\hat{T}) \cup \{u\})$ of non-bottleneck vertices there exists a path P from r to u such that $|P \cap X| \le |X|/2$.*

Proof. Assume that $u \in B_x$ and let $\hat{P} = x_0 \dots x_\ell$ be a path from $x_0 = r$ to $x_\ell = x$ in $\hat{T}$. Let further $F_0, \dots, F_\ell$ be the fins of $\hat{P}$ in $\hat{T}$. We partition the set X into $X_1, \dots, X_\ell$ where $X_i = X \cap F_i$ for $0 \le i \le \ell$. Lemma 19 allows us to construct the path P iteratively: any path that leads to u will pass through bottlenecks x_i, x_{i+1} in succession and visit only diblocks associated with F_i in the process. Since there are two internally vertex-disjoint paths between x_i, x_{i+1} for $1 \le i \le \ell$, we can always choose the path that has the smaller intersection with X_i. Stringing these paths together, we obtain the claimed path P. ◀

We want to argue that one of the following cases must hold: either the cut decomposition has bounded height and we can 're-root' any out-tree with many leaves into a rooted out-tree with a comparable number of leaves, or we can directly construct a rooted out-tree with many leaves. In both cases we apply Lemmas 4 and 9 to conclude that the instance has a solution. This approach has one obstacle: internal diblocks of the decomposition that contain only two vertices.

▶ **Definition 21** (Degenerate diblocks)**.** Let $\{B_x\}_{x\in\hat{T}}$ be the cut decomposition rooted at s. We call a diblock B_x *degenerate* if x is an internal node of $\hat{T}$ and $|B_x| = 2$.

Let us first convince ourselves that a long enough sequence of non-degenerate diblocks provides us with a rooted out-branching with many leaves.

▶ **Lemma 22.** *Let $(\hat{T}, \{B_x\}_{x\in\hat{T}})$ be the cut decomposition rooted at s of H and let y be a node in $\hat{T}$ such that the path $\hat{P}_{sy}$ from s to y in $\hat{T}$ contains at least ℓ nodes whose diblocks are non-degenerate. Then H contains an out-tree rooted at s with at least ℓ leaves.*

Proof. We construct an s-rooted out-tree T by repeated application of Lemma 12. Let $x_1, \ldots, x_\ell$ be a sequence of nodes in $\hat{P}_{sy}$ whose diblocks are non-degenerate, and for each $1 \leq i < \ell$ let x_i^+ be the node after x_i in $\hat{P}_{sy}$. We construct a sequence of s-rooted out-trees $T_1, \ldots, T_\ell$ such that for $1 \leq i \leq \ell$, the vertex x_i is a leaf of T_i, and T_i contains i leaves. First construct T_1 as a path from s to x_1, then for every $1 \leq i < \ell$ we construct an out-tree T_{i+1} from T_i as follows. Let $v_i \in B_{x_i} \setminus \{x_i, x_i^+\}$, which exists since B_{x_i} is non-degenerate, and let $P_{x_i x_i^+}$, $P_{x_i v_i}$ be a pair of paths in $H[B^*_{x_i}]$ from x_i to x_i^+ and to v_i respectively, which intersect only in x_i. Such paths exist by Lemma 12, and since x_i is a leaf of T_i, Lemma 17 implies that T_i is disjoint from $B^*_{x_i} \setminus \{x_i\}$. Hence the paths can be appended to T_i to form a new r-rooted out-tree T_{i+1} in H which contains a leaf in every diblock B_{x_j}, $1 \leq i$. Finally, note that the final tree T_ℓ contains two leaves in $B_{x_{\ell-1}}$, hence T_ℓ is an r-rooted out-tree with ℓ leaves. ◀

The next lemma is the last assertion that we will use to prove shortly that Rooted k-Distinct Branchings is FPT for digraphs D whose cut decomposition rooted at s contains no degenerate diblocks.

▶ **Lemma 23.** *Let $(\hat{T}, \{B_x\}_{x \in \hat{T}})$ be the cut decomposition of D rooted at s such that $\hat{T}$ is of height d and let T be an out-tree rooted at some vertex r with ℓ leaves. Then we can construct an out-tree T_s rooted at s with at least $(\ell - d)/2$ leaves.*

Proof. Assume that r is contained in the diblock B_x of the decomposition and let $x_p \ldots x_1 = \hat{P}_{sx}$ be a path from $s = x_p$ to $x = x_1$ in $\hat{T}$. Let L be the leaves of T and let $L' := L \setminus \hat{P}_{sx}$. Clearly, $|L'| \geq \ell - d$. Applying Corollary 20 with $X = L'$ and $u = r$, we obtain a path P_{sr} in D from s to r that avoids half of L'. We construct T_s in a similar fashion to the proof of Lemma 11. We begin with $T_s = P_{sr}$, then for every leaf $v \in L' \setminus P_{sr}$, proceed as follows: let P_v be the unique path from r to v in T and let P'_v be the segment of P_v from the last vertex x of P_v contained in T_s. Add all arcs and vertices of P'_v to T_s. Since P_v and thus P'_v contains no leaf of L' other than v, in the end of the process, all vertices of $L' \setminus P_{sr}$ are leaves of T_s. Since $|L' \setminus P_{sr}| \geq |L'|/2$, we conclude that T_s contains at least $(\ell - d)/2$ leaves, as claimed. ◀

The next lemma demonstrates that using Lemma 23 and a number of other results we can prove that if the height d of the cut decomposition of D is upper-bounded by a function in k, then Rooted k-Distinct Branchings on D is FPT. This shows that to prove that Rooted k-Distinct Branchings in general it suffices to consider separately the cases of bounded d and unbounded d. To provide an appropriate bound on d we will use further results on degenerate diblocks proved in Section 4.3.

▶ **Lemma 24.** *Let $(\hat{T}, \{B_x\}_{x \in \hat{T}})$ the cut decomposition rooted at s of height d. If $d \leq d(k)$ for some function $d(k) = \Omega(k)$ of k only, then we can solve* Rooted k-Distinct Branchings *on D in time $O^*(2^{O(d(k) \log^2 d(k))})$.*

Proof. By Theorem 2, in time $O^*(2^{O(d(k))})$ we can decide whether D has an out-branching with at least $2k + 2 + d(k)$ leaves. If D has such an out-branching, then by Lemma 23 D has a rooted out-tree with at least $k + 1$ leaves. This out-tree can be extended to a rooted out-branching with at least $k + 1$ leaves by Lemma 9. So by Lemma 4, (D, s, t) is a positive instance if and only if D has a rooted in-branching, which can be decided in polynomial time.

If D has no out-branching with at least $2k+2+d(k)$ leaves, by Theorem 1 the pathwidth of D is $O(d(k) \log d(k))$ and thus by Lemma 5 we can solve Rooted k-Distinct Branchings on D in time $O^*(2^{O(d(k) \log^2 d(k))})$. (Note that for the dynamic programming algorithm of

Lemma 5 we may fix roots of all out-branchings and all in-branchings of D by adding arcs $s's$ and tt' to D, where s' and t' are new vertices.) ◀

4.3 Handling degenerate diblocks

A full version of this section can be found in the complete version of the paper. Here is a key notion for our study of degenerate diblocks.

▶ **Definition 25** (Degenerate paths)**.** Let $(\hat{T}, \{B_x\}_{x\in\hat{T}})$ be a cut decomposition of D. We call a path $\hat{P}$ in $\hat{T}$ *monotone* if it is a subpath of a path from the root of $\hat{T}$ to some leaf of $\hat{T}$. We call a path $\hat{P}$ in $\hat{T}$ *degenerate* if it is monotone and every diblock B_x, $x \in \hat{P}$ is degenerate.

Let (D, s, t) be a strongly connected reduced instance of Rooted k-Distinct Branchings. As observed in Section 4.1, we can verify in polynomial time whether an arc participates in *some* rooted in- or out-branching. Let $R_s \subseteq A(D)$ be those arcs that do not participate in any rooted out-branching and $R_t \subseteq A(D)$ those that do not participate in any rooted in-branching. Since (D, s, t) is a reduced instance, we necessarily have that $R_s \cap R_t = \emptyset$.

▶ **Lemma 26.** *Let $\hat{P} = x_1 \dots x_\ell$ be a degenerate path of $(\hat{T}, \{B_x\}_{x\in\hat{T}})$ of D rooted at s. Then the following properties hold: every rooted out-branching contains $A(\hat{P})$, every arc x_jx_i with $j > i$ is contained in R_s, and there is no arc from x_i $(i < \ell)$ to B_y in D, where y is a descendant of x_i on $\hat{T}$, except for the arc x_ix_{i+1}.*

Let us fix a single degenerate path $\hat{P} = x_1 \dots x_\ell$. We categorize the arcs incident to $\hat{P}$ as follows: let A^+ contain all 'upward arcs' that originate in $\hat{P}$ and end in some diblock B_y where y is an ancestor of x_1, let A^0 contain all 'on-path arcs' x_jx_i, $j > i$, and let A^- contain all 'arcs from below' that originate from some diblock B_y where y is a (not necessarily proper) descendant of x_ℓ. By the lemma above this categorization is complete.

We will need the following reduction rules for (D, s, t):

Reduction Rule 1: If there are two arcs $x_iu, x_ju \in A^+ \cap R_t$ with $i < j$, remove x_ju.

Reduction Rule 2: If $\hat{P}[x, y] \subseteq \hat{P}$ is such that no vertex in $\hat{P}[x, y]$ is a tail of arcs in $A^+ \cup A^0$, contract $\hat{P}[x, y]$ into a single vertex.

Now we can state the main lemma of the section which finally enables us to proof the main result of this paper.

▶ **Lemma 27.** *Let $\hat{P}$ be a degenerate path in an instance reduced with respect to Rules 1 and 2. If $t \notin \hat{P}$, then $|\hat{P}| \leq 14k + 3$. Otherwise, $|\hat{P}| \leq 28k + 7$.*

Proof of Theorem 7. By Lemma 8, we may assume that D is strongly connected. Consider the longest monotone path $\hat{P}$ of $\hat{T}$. By Lemma 22, if $\hat{P}$ has at least $k + 1$ non- degenerate diblocks, then D has a rooted out-tree with at least $k + 1$ leaves. This out-tree can be extended to a rooted out-branching with at least $k+1$ leaves by Lemma 9. Thus, by Lemma 4, (D, s, t) is a positive instance if and only if D has a rooted in-branching, which can be decided in polynomial time.

Now assume that $\hat{P}$ has at most k non-degenerate diblocks. By Lemma 27 we may assume that before, between and after the non-degenerate diblocks there are $O(k)$ degenerate diblocks. Thus, the height of $\hat{T}$ is $O(k^2)$. Therefore, by Lemma 24, the time complexity for Theorem 7 is $O^*(2^{O(k^2 \log^2 k)})$. ◀

5 Conclusion

We showed that the Rooted k-Distinct Branchings problem is FPT for general digraphs parameterized by k, thereby settling open question of Bang-Jensen *et al.* [6]. The solution in particular uses a new digraph decomposition, the *rooted cut decomposition*, that we believe might be useful for settling other problems as well. We did not try to optimize the running time of the algorithm of Theorem 7. Perhaps, a more careful handling of degenerate diblocks may lead to an algorithm of running time $O^*(2^{O(k \log^2 k)})$.

References

1 N. Alon, F. V. Fomin, G. Gutin, M. Krivelevich, and S. Saurabh. Spanning directed trees with many leaves. *SIAM Journal on Discrete Mathematics*, 23(1):466–476, 2009.

2 J. Bang-Jensen. Edge-disjoint in- and out-branchings in tournaments and related path problems. *Journal of Combinatorial Theory, Series B*, 51(1):1–23, 1991.

3 J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2002.

4 J. Bang-Jensen and J. Huang. Quasi-transitive digraphs. *Journal of Graph Theory*, 20:141–161, 1995.

5 J. Bang-Jensen and J. Huang. Arc-disjoint in- and out-branchings with the same root in locally semicomplete digraphs. *Journal of Graph Theory*, 77:278–298, 2014.

6 J. Bang-Jensen, S. Saurabh, and S. Simonsen. Parameterized algorithms for non-separating trees and branchings in digraphs. *Algorithmica*, 76(1):279–296, 2016.

7 J. Bang-Jensen and S. Simonsen. Arc-disjoint paths and trees in 2-regular digraphs. *Discrete Applied Mathematics*, 161:2724–2730, 2013.

8 J. Bang-Jensen, S. Thomassé, and A. Yeo. Small degree out-branchings. *Journal of Graph Theory*, 42(4):297–307, 2003.

9 J. Bang-Jensen and A. Yeo. The minimum spanning strong subdigraph problem is fixed parameter tractable. *Discrete Applied Mathematics*, 156:2924–2929, 2008.

10 K. Bérczi, S. Fujishige, and N. Kamiyama. A linear-time algorithm to find a pair of arc-disjoint spanning in-arborescence and out-arborescence in a directed acyclic graph. *Information Processing Letters*, 109(23-24):1227–1231, 2009.

11 M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

12 J. Daligault, G. Gutin, E. J. Kim, and A. Yeo. FPT algorithms and kernels for the directed k-leaf problem. *Journal of Computer and System Sciences*, 76(2):144–152, 2010.

13 R.G. Downey and M.R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

14 J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards, Section B*, 71B:233–240, 1967.

15 J. Edmonds. Edge-disjoint branchings. In B. Rustin, editor, *Combinatorial Algorithms*, pages 91–96. Academic Press, 1973.

16 A. Fradkin and P.D. Seymour. Tournament pathwidth and topological containment. *Journal of Combinatorial Theory, Series B*, 103(3):374–384, 2013.

17 K. Kawarabayashi and S. Kreutzer. The directed grid theorem. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 655–664, 2015.

18 I. Kim and P. D. Seymour. Tournament minors. *Journal of Combinatorial Theory, Series B*, 112:138–153, 2015.

19 J. Kneis, A. Langer, and P. Rossmanith. A new algorithm for finding trees with many leaves. *Algorithmica*, 61(4):882–897, 2011.

Fast Regression with an ℓ_∞ Guarantee*

Eric Price[1], Zhao Song[2], and David P. Woodruff[3]

1 **Dept. of Computer Science, The University of Texas at Austin, Austin, USA**
ecprice@cs.utexas.edu
2 **Dept. of Computer Science, The University of Texas at Austin, Austin, USA**
zhaos@utexas.edu
3 **IBM Research Almaden, San Jose, CA, USA**
dpwoodru@us.ibm.com

Abstract

Sketching has emerged as a powerful technique for speeding up problems in numerical linear algebra, such as regression. In the overconstrained regression problem, one is given an $n \times d$ matrix A, with $n \gg d$, as well as an $n \times 1$ vector b, and one wants to find a vector $\hat{x}$ so as to minimize the residual error $\|Ax - b\|_2$. Using the sketch and solve paradigm, one first computes $S \cdot A$ and $S \cdot b$ for a randomly chosen matrix S, then outputs $x' = (SA)^\dagger Sb$ so as to minimize $\|SAx' - Sb\|_2$.

The sketch-and-solve paradigm gives a bound on $\|x' - x^*\|_2$ when A is well-conditioned. Our main result is that, when S is the subsampled randomized Fourier/Hadamard transform, the error $x' - x^*$ behaves as if it lies in a "random" direction within this bound: for any fixed direction $a \in \mathbb{R}^d$, we have with $1 - d^{-c}$ probability that

$$\langle a, x' - x^* \rangle \lesssim \frac{\|a\|_2 \|x' - x^*\|_2}{d^{\frac{1}{2}-\gamma}}, \tag{1}$$

where $c, \gamma > 0$ are arbitrary constants. This implies $\|x' - x^*\|_\infty$ is a factor $d^{\frac{1}{2}-\gamma}$ smaller than $\|x' - x^*\|_2$. It also gives a better bound on the generalization of x' to new examples: if rows of A correspond to examples and columns to features, then our result gives a better bound for the error introduced by sketch-and-solve when classifying fresh examples. We show that not all oblivious subspace embeddings S satisfy these properties. In particular, we give counterexamples showing that matrices based on Count-Sketch or leverage score sampling do not satisfy these properties.

We also provide lower bounds, both on how small $\|x' - x^*\|_2$ can be, and for our new guarantee (1), showing that the subsampled randomized Fourier/Hadamard transform is nearly optimal. Our lower bound on $\|x' - x^*\|_2$ shows that there is an $O(1/\epsilon)$ separation in the dimension of the optimal oblivious subspace embedding required for outputting an x' for which $\|x' - x^*\|_2 \leq \epsilon \|Ax^* - b\|_2 \cdot \|A^\dagger\|_2$, compared to the dimension of the optimal oblivious subspace embedding required for outputting an x' for which $\|Ax' - b\|_2 \leq (1+\epsilon)\|Ax^* - b\|_2$, that is, the former problem requires dimension $\Omega(d/\epsilon^2)$ while the latter problem can be solved with dimension $O(d/\epsilon)$. This explains the reason known upper bounds on the dimensions of these two variants of regression have differed in prior work.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems

Keywords and phrases Linear regression, Count-Sketch, Gaussians, Leverage scores, ℓ_∞-guarantee

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.59

* A full version of the paper is available at https://arxiv.org/abs/1705.10723.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 59; pp. 59:1–59:14

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

1 Introduction

Oblivious subspace embeddings (OSEs) were introduced by Sarlos [23] to solve linear algebra problems more quickly than traditional methods. An OSE is a distribution of matrices $S \in \mathbb{R}^{m\times n}$ with $m \ll n$ such that, for any d-dimensional subspace $U \subset \mathbb{R}^n$, with "high" probability S preserves the norm of every vector in the subspace. OSEs are a generalization of the classic Johnson-Lindenstrauss lemma from vectors to subspaces. Formally, we require that with probability $1-\delta$,

$$\|Sx\|_2 = (1\pm\epsilon)\|x\|_2$$

simultaneously for all $x \in U$, that is, $(1-\epsilon)\|x\|_2 \le \|Sx\|_2 \le (1+\epsilon)\|x\|_2$.

A major application of OSEs is to regression. The regression problem is, given $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times d}$ for $n \ge d$, to solve for

$$x^* = \arg\min_{x\in\mathbb{R}^d} \|Ax-b\|_2. \tag{2}$$

Because A is a "tall" matrix with more rows than columns, the system is overdetermined and there is likely no solution to $Ax = b$, but regression will find the closest point to b in the space spanned by A. The classic answer to regression is to use the Moore-Penrose pseudoinverse: $x^* = A^\dagger b$ where

$$A^\dagger = (A^\top A)^{-1}A^\top$$

is the "pseudoinverse" of A (assuming A has full column rank, which we will typically do for simplicity). This classic solution takes $O(nd^{\omega-1}+d^\omega)$ time, where $\omega < 2.373$ is the matrix multiplication constant [9, 25, 12]: $nd^{\omega-1}$ time to compute $A^\top A$ and d^ω time to compute the inverse.

OSEs speed up the process by replacing (2) with

$$x' = \arg\min_x \|SAx - Sb\|_2$$

for an OSE S on $d+1$-dimensional spaces. This replaces the $n\times d$ regression problem with an $m\times d$ problem, which can be solved more quickly since $m \ll n$. Because $Ax-b$ lies in the $d+1$-dimensional space spanned by b and the columns of A, with high probability S preserves the norm of $SAx-Sb$ to $1\pm\epsilon$ for all x. Thus,

$$\|Ax'-b\|_2 \le \frac{1+\epsilon}{1-\epsilon}\|Ax^*-b\|_2.$$

That is, S produces a solution x' which preserves the *cost* of the regression problem. The running time for this method depends on (1) the reduced dimension m and (2) the time it takes to multiply S by A. We can compute these for "standard" OSE types:

- If S has i.i.d. Gaussian entries, then $m = O(d/\epsilon^2)$ is sufficient (and in fact, $m \ge d/\epsilon^2$ is required [20]). However, computing SA takes $O(mnd) = O(nd^2/\epsilon^2)$ time, which is worse than solving the original regression problem (one can speed this up using fast matrix multiplication, though it is still worse than solving the original problem).
- If S is a subsampled randomized Hadamard transform (SRHT) matrix with random sign flips (see Theorem 2.4 in [26] for a survey, and also see [8] which gives a recent improvement) then m increases to $\widetilde{O}(d/\epsilon^2\cdot\log n)$, where $\widetilde{O}(f) = f\,\mathrm{poly}(\log(f))$. But now, we can compute SA using the fast Hadamard transform in $O(nd\log n)$ time. This makes the overall regression problem take $O(nd\log n + d^\omega/\epsilon^2)$ time.

- If S is a random sparse matrix with random signs (the "Count-Sketch" matrix), then $m = d^{1+\gamma}/\epsilon^2$ suffices for $\gamma > 0$ a decreasing function of the sparsity [5, 18, 19, 3, 6]. (The definition of a Count-Sketch matrix is, for any $s \geq 1$, $S_{i,j} \in \{0, -1/\sqrt{s}, 1/\sqrt{s}\}$, $\forall i \in [m], j \in [n]$ and the column sparsity of matrix S is s. Independently in each column s positions are chosen uniformly at random without replacement, and each chosen position is set to $-1/\sqrt{s}$ with probability 1/2, and $+1/\sqrt{s}$ with probability 1/2.) Sparse OSEs can benefit from the sparsity of A, allowing for a running time of $\widetilde{O}(\text{nnz}(A)) + \widetilde{O}(d^{\omega}/\epsilon^2)$, where $\text{nnz}(A)$ denotes the number of non-zeros in A.

When n is large, the latter two algorithms are substantially faster than the naïve $nd^{\omega-1}$ method.

1.1 Our Contributions

Despite the success of using subspace embeddings to speed up regression, often what practitioners are interested is not in preserving the cost of the regression problem, but rather in the *generalization* or *prediction* error provided by the vector x'. Ideally, we would like for any future (unseen) example $a \in \mathbb{R}^d$, that $\langle a, x' \rangle \approx \langle a, x^* \rangle$ with high probability.

Ultimately one may want to use x' to do classification, such as regularized least squares classification (RLSC) [22], which has been found in cases to do as well as support vector machines but is much simpler [27]. In this application, given a training set of examples with multiple (non-binary) labels identified with the rows of an $n \times d$ matrix A, one creates an $n \times r$ matrix B, each column indicating the presence or absence of one of the r possible labels in each example. One then solves the multiple response regression problem $\min_X \|AX - B\|_F$, and uses X to classify future examples. A commonly used method is for a future example a, to compute $\langle a, x_1 \rangle, \ldots, \langle a, x_r \rangle$, where $x_1, \ldots, x_r$ are the columns of X. One then chooses the label i for which $\langle a, x_i \rangle$ is maximum.

For this to work, we would like the inner products $\langle a, x'_1 \rangle, \ldots, \langle a, x'_r \rangle$ to be close to $\langle a, x^*_1 \rangle, \ldots, \langle a, x^*_r \rangle$, where X' is the solution to $\min_X \|SAX - SB\|_F$ and X^* is the solution to $\min_X \|AX - B\|_F$. For any $O(1)$-accurate OSE on $d + r$ dimensional spaces [23], which also satisfies so-called approximate matrix multiplication with error $\epsilon' = \epsilon/\sqrt{(d+r)}$, we get that

$$\|x' - x^*\|_2 \leq O(\epsilon) \cdot \|Ax^* - b\|_2 \cdot \|A^\dagger\|_2 \tag{3}$$

where $\|A^\dagger\|$ is the spectral norm of $A^\dagger$, which equals the reciprocal of the smallest singular value of A. To obtain a generalization error bound for an unseen example a, one has

$$|\langle a, x^* \rangle - \langle a, x' \rangle| = |\langle a, x^* - x' \rangle| \leq \|x^* - x'\|_2 \|a\|_2 = O(\epsilon)\|a\|_2 \|Ax^* - b\|_2 \|A^\dagger\|_2, \tag{4}$$

which could be tight if given only the guarantee in (3). However, if the difference vector $x' - x^*$ were distributed in a uniformly random direction subject to (3), then one would expect an $\widetilde{O}(\sqrt{d})$ factor improvement in the bound. This is what our main theorem shows:

▶ **Theorem 1** (Main Theorem, informal). *Suppose $n \leq \text{poly}(d)$. Let S be a subsampled randomized Hadamard transform matrix with $m = d^{1+\gamma}/\epsilon^2$ rows for an arbitrarily small constant $\gamma > 0$. For $x' = \arg\min_x \|SAx - Sb\|_2$ and $x^* = \arg\min_x \|Ax - b\|_2$, and any fixed $a \in \mathbb{R}^d$,*

$$|\langle a, x^* \rangle - \langle a, x' \rangle| \leq \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|Ax^* - b\|_2 \|A^\dagger\|_2. \tag{5}$$

with probability $1 - 1/d^C$ for an arbitrarily large constant $C > 0$. This implies that

$$\|x^* - x'\|_\infty \leq \frac{\epsilon}{\sqrt{d}} \|Ax^* - b\|_2 \|A^\dagger\|_2 \tag{6}$$

with $1 - 1/d^{C-1}$ probability.

If $n > \text{poly}(d)$, then by first composing S with a Count-Sketch OSE with $\text{poly}(d)$ rows, one can achieve the same guarantee.

(Here γ is a constant going to zero as n increases, see Theorem 10 for a formal statement of Theorem 1).

Notice that Theorem 1 is considerably stronger than that of (4) provided by existing guarantees. Indeed, in order to achieve the guarantee (6) in Theorem 1, one would need to set $\epsilon' = \epsilon/\sqrt{d}$ in existing OSEs, resulting in $\Omega(d^2/\epsilon^2)$ rows. In contrast, we achieve only $d^{1+\gamma}/\epsilon^2$ rows. We can improve the bound in Theorem 1 to $m = d/\epsilon^2$ if S is a matrix of i.i.d. Gaussians; however, as noted, computing $S \cdot A$ is slower in this case.

Note that Theorem 1 also *makes no distributional assumptions* on the data, and thus the data could be heavy-tailed or even adversarially corrupted. This implies that our bound is still useful when the rows of A are not sampled independently from a distribution with bounded variance.

The ℓ_∞ bound (6) of Theorem 1 is achieved by applying (5) to the standard basis vectors $a = e_i$ for each $i \in [d]$ and applying a union bound. This ℓ_∞ guarantee often has a more natural interpretation than the ℓ_2 guarantee – if we think of the regression as attributing the observable as a sum of various factors, (6) says that the contribution of each factor is estimated well. One may also see our contribution as giving a way for estimating the pseudoinverse $A^\dagger$ *entrywise*. Namely, we get that $(SA)^\dagger S \approx A^\dagger$ in the sense that each entry is within additive $O(\epsilon\sqrt{\frac{\log d}{d}}\|A^\dagger\|_2)$. There is a lot of work on computing entries of inverses of a matrix, see, e.g., [1, 16].

Another benefit of the ℓ_∞ guarantee is when the regression vector x^* is expected to be *k-sparse* (e.g. [14]). In such cases, thresholding to the top k entries will yield an ℓ_2 guarantee a factor $\sqrt{k/d}$ better than (3).

One could ask if Theorem 1 also holds for sparse OSEs, such as the Count-Sketch. Surprisingly, we show that one cannot achieve the generalization error guarantee in Theorem 1 with high probability, say, $1 - 1/d$, using such embeddings, despite the fact that such embeddings do approximate the cost of the regression problem up to a $1 + \epsilon$ factor with high probability. This shows that the generalization error guarantee is achieved by some subspace embeddings but not all.

▶ **Theorem 2** (Not all subspace embeddings give the ℓ_∞ guarantee)**.** *The Count-Sketch matrix with $d^{1.5}$ rows and sparsity $d^{.25}$ – which is an OSE with exponentially small failure probability – with constant probability will have a result x' that does not satisfy the ℓ_∞ guarantee (6).*

We can show that Theorem 1 holds for S based on the Count-Sketch OSE T with $d^{O(C)}/\epsilon^2$ rows with $1 - 1/d^C$ probability. We can thus compose the Count-Sketch OSE with the SRHT matrix and obtain an $O(\text{nnz}(A)) + \text{poly}(d/\epsilon)$ time algorithm to compute $S \cdot TA$ achieving (6). We can also compute $R \cdot S \cdot T \cdot A$, where R is a matrix of Gaussians, which is more efficient now that STA only has $d^{1+\gamma}/\epsilon^2$ rows; this will reduce the number of rows to d/ϵ^2.

Another common method of dimensionality reduction for linear regression is *leverage score sampling* [10, 15, 21, 7], which subsamples the rows of A by choosing each row with probability proportional to its "leverage scores". With $O(d\log(d/\delta)/\epsilon^2)$ rows taken, the result x' will satisfy the ℓ_2 bound (3) with probability $1 - \delta$. However, it does not give a good ℓ_∞ bound:

▶ **Theorem 3** (Leverage score sampling does not give the ℓ_∞ guarantee)**.** *Leverage score sampling with $d^{1.5}$ rows – which satisfies the ℓ_2 bound with exponentially small failure probability – with constant probability will have a result x' that does not satisfy the ℓ_∞ guarantee (6).*

Finally, we show that the $d^{1+\gamma}/\epsilon^2$ rows that SRHT matrices use is roughly optimal:

▶ **Theorem 4** (Lower bounds for ℓ_2 and ℓ_∞ guarantees). *Any sketching matrix distribution over $m \times n$ matrices that satisfies either the ℓ_2 guarantee* (3) *or the ℓ_∞ guarantee* (6) *must have $m \gtrsim \min(n, d/\epsilon^2)$.*

Notice that our result shows the necessity of the $1/\epsilon$ separation between the results originally defined in Equation (3) and (4) of Theorem 12 of [23]. If we want to output some vector x' such that $\|Ax' - b\|_2 \leq (1+\epsilon)\|Ax^* - b\|_2$, then it is known that $m = \Theta(d/\epsilon)$ is necessary and sufficient. However, if we want to output a vector x' such that $\|x' - x^*\|_2 \leq \epsilon\|Ax^* - b\|_2 \cdot \|A^\dagger\|_2$, then we show that $m = \Theta(d/\epsilon^2)$ is necessary and sufficient.

1.1.1 Comparison to Gradient Descent

While this work is primarily about sketching methods, one could instead apply iterative methods such as gradient descent, after appropriately preconditioning the matrix, see, e.g., [2, 28, 5]. That is, one can use an OSE with constant ϵ to construct a preconditioner for A and then run conjugate gradient using the preconditioner. This gives an overall dependence of $\log(1/\epsilon)$.

The main drawback of this approach is that one loses the ability to save on storage space or number of passes when A appears in a stream, or to save on communication or rounds when A is distributed. Given increasingly large data sets, such scenarios are now quite common, see, e.g., [4] for regression algorithms in the data stream model. In situations where the entries of A appear sequentially, for example, a row at a time, one does not need to store the full $n \times d$ matrix A but only the $m \times d$ matrix SA.

Also, iterative methods can be less efficient when solving multiple response regression, where one wants to minimize $\|AX - B\|$ for a $d \times t$ matrix X and an $n \times t$ matrix B. This is the case when ϵ is constant and t is large, which can occur in some applications (though there are also other applications for which ϵ is very small). For example, conjugate gradient with a preconditioner will take $\widetilde{O}(ndt)$ time while using an OSE directly will take only $\widetilde{O}(nd + d^2t)$ time (since one effectively replaces n with $O(d)$ after computing $S \cdot A$), separating t from d. Multiple response regression, arises, for example, in the RLSC application above.

1.1.2 Proof Techniques

Theorem 1. As noted in Theorem 2, there are some OSEs for which our generalization error bound does not hold. This hints that our analysis is non-standard and cannot use generic properties of OSEs as a black box. Indeed, in our analysis, we have to consider matrix products of the form $S^\top S(UU^\top S^\top S)^k$ for our random sketching matrix S and a fixed matrix U, where k is a positive integer. We stress that it is the *same matrix* S appearing multiple times in this expression, which considerably complicates the analysis, and does not allow us to appeal to standard results on approximate matrix product (see, e.g., [26] for a survey). The key idea is to recursively reduce $S^\top S(UU^\top S^\top S)^k$ using a property of S. We use properties that only hold for specifics OSEs S: first, that each column of S is unit vector; and second, that for all pairs (i, j) and $i \neq j$, the inner product between S_i and S_j is at most $\sqrt{\log n}/\sqrt{m}$ with probability $1 - 1/\operatorname{poly}(n)$.

Theorems 2 and 3. To show that Count-Sketch does not give the ℓ_∞ guarantee, we construct a matrix A and vector b as in Figure 1, which has optimal solution x^* with all coordinates $1/\sqrt{d}$. We then show, for our setting of parameters, that there likely exists an

index $j \in [d]$ satisfying the following property: the jth column of S has disjoint support from the kth column of S for all $k \in [d+\alpha] \setminus \{j\}$ except for a single $k > d$, for which S_j and S_k share exactly one common entry in their support. In such cases we can compute x'_j explicitly, getting $|x'_j - x^*_j| = \frac{1}{s\sqrt{\alpha}}$. By choosing suitable parameters in our construction, this gives that $\|x' - x^*\|_\infty \gg \frac{1}{\sqrt{d}}$. The lower bound for leverage score sampling follows a similar construction.

Theorem 4. The lower bound proof for the ℓ_2 guarantee uses Yao's minimax principle. We are allowed to fix an $m \times n$ sketching matrix S and design a distribution over $[A\ b]$. We first write the sketching matrix $S = U\Sigma V^\top$ in its singular value decomposition (SVD). We choose the $d+1$ columns of the adjoined matrix $[A, b]$ to be random orthonormal vectors. Consider an $n \times n$ orthonormal matrix R which contains the columns of V as its first m columns, and is completed on its remaining $n - m$ columns to an arbitrary orthonormal basis. Then $S \cdot [A, b] = V^\top R R^\top \cdot [A, b] = [U\Sigma I_m, 0] \cdot [R^\top A, R^\top b]$. Notice that $[R^\top A, R^\top b]$ is equal in distribution to $[A, b]$, since R is fixed and $[A, b]$ is a random matrix with $d+1$ orthonormal columns. Therefore, $S \cdot [A, b]$ is equal in distribution to $[U\Sigma G, U\Sigma h]$ where $[G, h]$ corresponds to the first m rows of an $n \times (d+1)$ uniformly random matrix with orthonormal columns.

A key idea is that if $n = \Omega(\max(m, d)^2)$, then by a result of Jiang [13], any $m \times (d+1)$ submatrix of a random $n \times n$ orthonormal matrix has $o(1)$ total variation distance to a $d \times d$ matrix of i.i.d. $N(0, 1/n)$ random variables, and so any events that would have occurred had G and h been independent i.i.d. Gaussians, occur with the same probability for our distribution up to an $1 - o(1)$ factor, so we can assume G and h are independent i.i.d. Gaussians in the analysis.

The optimal solution x' in the sketch space equals $(SA)^\dagger Sb$, and by using that SA has the form $U\Sigma G$, one can manipulate $\|(SA)^\dagger Sb\|$ to be of the form $\|\tilde{\Sigma}^\dagger (\Sigma R)^\dagger \Sigma h\|_2$, where the SVD of G is $R\tilde{\Sigma}T$. We can upper bound $\|\tilde{\Sigma}\|_2$ by $\sqrt{r/n}$, since it is just the maximum singular value of a Gaussian matrix, where r is the rank of S, which allows us to lower bound $\|\tilde{\Sigma}^\dagger (\Sigma R)^\dagger \Sigma h\|_2$ by $\sqrt{n/r}\|(\Sigma R)^\dagger \Sigma h\|_2$. Then, since h is i.i.d. Gaussian, this quantity concentrates to $\frac{1}{\sqrt{r}}\|(\Sigma R)^\dagger \Sigma h\|$, since $\|Ch\|^2 \approx \|C\|_F^2/n$ for a vector h of i.i.d. $N(0, 1/n)$ random variables. Finally, we can lower bound $\|(\Sigma R)^\dagger \Sigma\|_F^2$ by $\|(\Sigma R)^\dagger \Sigma R R^\top\|_F^2$ by the Pythagorean theorem, and now we have that $(\Sigma R)^\dagger \Sigma R$ is the identity, and so this expression is just equal to the rank of ΣR, which we prove is at least d. Noting that $x^* = 0$ for our instance, putting these bounds together gives $\|x' - x^*\| \geq \sqrt{d/r}$. The last ingredient is a way to ensure that the rank of S is at least d. Here we choose another distribution on inputs A and b for which it is trivial to show the rank of S is at least d with large probability. We require S be good on the mixture. Since S is fixed and good on the mixture, it is good for both distributions individually, which implies we can assume S has rank d in our analysis of the first distribution above.

Notation. For a positive integer, let $[n] = \{1, 2, \ldots, n\}$. For a vector $x \in \mathbb{R}^n$, define $\|x\|_2 = (\sum_{i=1}^n x_i^2)^{\frac{1}{2}}$ and $\|x\|_\infty = \max_{i \in [n]} |x_i|$. For a matrix $A \in \mathbb{R}^{m \times n}$, define $\|A\|_2 = \sup_x \|Ax\|_2 / \|x\|_2$ to be the spectral norm of A and $\|A\|_F = (\sum_{i,j} A_{i,j}^2)^{1/2}$ to be the Frobenius norm of A. We use $A^\dagger$ to denote the Moore-Penrose pseudoinverse of $m \times n$ matrix A, which if $A = U\Sigma V^\top$ is its SVD (where $U \in \mathbb{R}^{m \times n}$, $\Sigma \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{n \times n}$ for $m \geq n$), is given by $A^\dagger = V\Sigma^{-1}U^\top$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c, C.

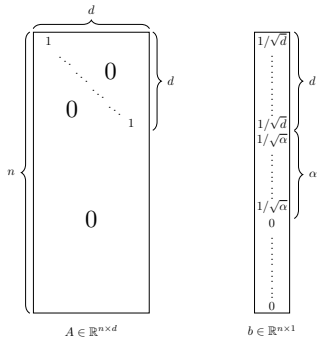


Figure 1 Our construction of A and b for the proof that Count-Sketch does not obey the ℓ_∞ guarantee. $\alpha < d$.

▶ **Definition 5** (Subspace Embedding). A $(1 \pm \epsilon)$ ℓ_2-subspace embedding for the column space of an $n \times d$ matrix A is a matrix S for which for all $x \in \mathbb{R}^d$, $\|SAx\|_2^2 = (1 \pm \epsilon)\|Ax\|_2^2$.

▶ **Definition 6** (Approximate Matrix Product). Let $0 < \epsilon < 1$ be a given approximation parameter. Given matrices A and B, where A and B each have n rows, the goal is to output a matrix C so that $\|A^\top B - C\|_F \leq \epsilon \|A\|_F \|B\|_F$. Typically C has the form $A^\top S^\top SB$, for a random matrix S with a small number of rows. In particular, this guarantee holds for the subsampled randomized Hadamard transform S with $O(\epsilon^{-2})$ rows [11].

Due to space constraints, several proofs are deferred to the full version of our paper.

2 Warmup: Gaussians OSEs

We first show that if S is a Gaussian random matrix, then it satisfies the generalization guarantee. This follows from the rotational invariance of the Gaussian distribution.

▶ **Theorem 7.** *Suppose $A \in \mathbb{R}^{n \times d}$ has full column rank. If the entries of $S \in \mathbb{R}^{m \times n}$ are i.i.d. $N(0, 1/m)$, $m = O(d/\epsilon^2)$, then for any vectors a, b and $x^* = A^\dagger b$, we have, with probability $1 - 1/\operatorname{poly}(d)$,*

$$|a^\top (SA)^\dagger Sb - a^\top x^*| \lesssim \frac{\epsilon\sqrt{\log d}}{\sqrt{d}} \|a\|_2 \|b - Ax^*\|_2 \|A^\dagger\|_2.$$

Because SA has full column rank with probability 1, $(SA)^\dagger SA = I$. Therefore

$$|a^\top (SA)^\dagger Sb - a^\top x^*| = |a^\top (SA)^\dagger S(b - Ax^*)| = |a^\top (SA)^\dagger S(b - AA^\dagger b)|.$$

Thus it suffices to only consider vectors b where $A^\dagger b = 0$, or equivalently $U^\top b = 0$. In such cases, SU will be independent of Sb, which will give the result. The proof is in the full version.

3 SRHT Matrices

We first provide the definition of the subsampled randomized Hadamard transform(SRHT): Let $S = \frac{1}{\sqrt{rn}} PH_n D$, where D is an $n \times n$ diagonal matrix with i.i.d. diagonal entries $D_{i,i}$, for which $D_{i,i}$ in uniform on $\{-1, +1\}$. Here H_n is the Hadamard matrix of size $n \times n$, and we assume n is a power of 2. Here, $H_n = [H_{n/2},\ H_{n/2}; H_{n/2},\ -H_{n/2}]$ and $H_1 = [1]$. The $r \times n$ matrix P samples r coordinates of an n dimensional vector uniformly at random.

For other subspace embeddings, we no longer have that SU and Sb are independent. To analyze them, we start with a claim that allows us to relate the inverse of a matrix to a power series.

▶ **Claim 8.** *Let $S \in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{n\times d}$ have SVD $A = U\Sigma V^\top$, and define $T \in \mathbb{R}^{d\times d}$ by $T = I_d - U^\top S^\top SU$. Suppose SA has linearly independent columns and $\|T\|_2 \le 1/2$. Then*

$$(SA)^\dagger S = V\Sigma^{-1}\left(\sum_{k=0}^{\infty} T^k\right)U^\top S^\top S. \tag{7}$$

Proof.

$$(SA)^\dagger S = (A^\top S^\top SA)^{-1}A^\top S^\top S = (V\Sigma U^\top S^\top SU\Sigma V^\top)^{-1}V\Sigma U^\top S^\top S$$
$$= V\Sigma^{-1}(U^\top S^\top SU)^{-1}U^\top S^\top S = V\Sigma^{-1}(I_d - T)^{-1}U^\top S^\top S = V\Sigma^{-1}(\sum_{k=0}^{\infty} T^k)U^\top S^\top S,$$

where in the last equality, since $\|T\|_2 < 1$, the von Neumann series $\sum_{k=0}^{\infty} T^k$ converges to $(I_d - T)^{-1}$. ◀

We then bound the kth term of this sum:

▶ **Lemma 9.** *Let $S \in \mathbb{R}^{r\times n}$ be the subsampled randomized Hadamard transform, and let a be a unit vector. Then with probability $1 - 1/\mathrm{poly}(n)$, we have*

$$|a^\top S^\top S(UU^\top S^\top S)^k b| = O(\log^k n)\cdot(O(d(\log n)/r) + 1)^{\frac{k-1}{2}}$$
$$\cdot(\sqrt{d}\|b\|_2(\log n)/r + \|b\|_2(\log^{\frac{1}{2}} n)/r^{\frac{1}{2}}).$$

Hence, for r at least $d\log^{2k+2} n\log^2(n/\epsilon)/\epsilon^2)$, this is at most $O(\|b\|_2\epsilon/\sqrt{d})$.

We defer the proof of this lemma to the next section, and now show how the lemma lets us prove that SRHT matrices satisfy the generalization bound with high probability:

▶ **Theorem 10.** *Suppose $A \in \mathbb{R}^{n\times d}$ has full column rank with $\log n = d^{o(1)}$. Let $S \in \mathbb{R}^{m\times n}$ be a subsampled randomized Hadamard transform with $m = O(d^{1+\alpha}/\epsilon^2)$ for $\alpha = \Theta(\sqrt{\frac{\log\log n}{\log d}})$. For any vectors a, b and $x^* = A^\dagger b$, we have*

$$|a^\top(SA)^\dagger Sb - a^\top x^*| \lesssim \frac{\epsilon}{\sqrt{d}}\|a\|_2\|b - Ax^*\|_2\|\Sigma^{-1}\|_2$$

with probability $1 - 1/\mathrm{poly}(d)$.

Proof. Define $\Delta = \Theta\left(\frac{1}{\sqrt{m}}\right)(\log^c d)\|a\|_2\|b - Ax^*\|_2\|\Sigma^{-1}\|_2$. For a constant $c > 0$, we have that S is a $(1\pm\gamma)$ ℓ_2-subspace embedding (Definition 5) for $\gamma = \sqrt{\frac{d\log^c n}{m}}$ with probability $1 - 1/\mathrm{poly}(d)$ (see, e.g., Theorem 2.4 of [26] and references therein), so $\|SUx\|_2 = (1\pm\gamma)\|Ux\|_2$ for all x, which we condition on. Hence for $T = I_d - U^\top S^\top SU$, we have $\|T\|_2 \le (1+\gamma)^2 - 1 \lesssim \gamma$. In particular, $\|T\|_2 < 1/2$ and we can apply Claim 8.

As in Section 2, SA has full column rank if S is a subspace embedding, so $(SA)^\dagger SA = I$ and we may assume $x^* = 0$ without loss of generality.

By the approximate matrix product (Definition 6), we have for some c that

$$|a^\top V\Sigma^{-1}U^\top S^\top Sb| \le \frac{\log^c d}{\sqrt{m}}\|a\|_2\|b\|_2\|\Sigma^{-1}\|_2 \le \Delta, \tag{8}$$

with $1 - 1/\text{poly}(d)$ probability. Suppose this event occurs, bounding the $k = 0$ term of (7). Hence it suffices to show that the $k \geq 1$ terms of (7) are bounded by Δ.

By approximate matrix product, we also have with $1 - 1/d^2$ probability that

$$\|U^\top S^\top Sb\|_F \leq \frac{\log^c d}{\sqrt{m}}\|U^\top\|_F\|b\|_2 \leq \frac{\log^c d\sqrt{d}}{\sqrt{m}}\|b\|_2.$$

Combining with $\|T\|_2 \lesssim \gamma$ we have for any k that

$$|a^\top V\Sigma^{-1}T^kU^\top S^\top Sb| \lesssim \gamma^k(\log^c d)\frac{\sqrt{d}}{\sqrt{m}}\|a\|_2\|\Sigma^{-1}\|_2\|b\|_2.$$

Since this decays exponentially in k at a rate of $\gamma < 1/2$, the sum of all terms greater than k is bounded by the kth term. As long as

$$m \gtrsim \frac{1}{\epsilon^2}d^{1+\frac{1}{k}}\log^c n, \tag{9}$$

we have $\gamma = \sqrt{\frac{d\log^c n}{m}} < \epsilon d^{-1/(2k)}/\log^c n$, so that

$$\sum_{k'\geq k}|a^\top V\Sigma^{-1}T^{k'}U^\top S^\top Sb| \lesssim \frac{\epsilon}{\sqrt{d}}\|a\|_2\|\Sigma^{-1}\|_2\|b\|_2.$$

On the other hand, by Lemma 9 (increasing m by a C^k factor) we have for all k that

$$|a^\top V^\top\Sigma^{-1}U^\top S^\top S(UU^\top S^\top S)^kb| \lesssim \frac{1}{2^k}\frac{\epsilon}{\sqrt{d}}\|a\|_2\|b\|_2\|\Sigma^{-1}\|_2,$$

with probability at least $1 - 1/\text{poly}(d)$, as long as $m \gtrsim d(C\log n)^{2k+2}\log^2(n/\epsilon)/\epsilon^2$, for a sufficiently large constant C. Since the T^k term can be expanded as a sum of 2^k terms of this form, we get that

$$\sum_{k'=1}^{k}|a^\top V\Sigma^{-1}T^{k'}U^\top S^\top Sb| \lesssim \frac{\epsilon}{\sqrt{d}}\|a\|_2\|b\|_2\|\Sigma^{-1}\|_2,$$

with probability at least $1 - 1/\text{poly}(d)$, as long as $m \gtrsim d(C\log n)^{2k+2}\log^2(n/\epsilon)/\epsilon^2$ for a sufficiently large constant C. Combining with (9), the result holds as long as $m \gtrsim \epsilon^{-2}d\log^c n\max((C\log n)^{2k+2}, d^{\frac{1}{k}})$, for any k. Setting $k = \Theta(\sqrt{\frac{\log d}{\log\log n}})$ gives the result. ◀

Combining Different Matrices. In some cases it can make sense to combine different sketching matrices that satisfy the generalization bound. We defer the details to the full version.

▶ **Theorem 11.** *Let $A \in \mathbb{R}^{n\times d}$, and let $R \in \mathbb{R}^{m\times r}$ and $S \in \mathbb{R}^{r\times n}$ be drawn from distributions of matrices that are ϵ-approximate OSEs and satisfy the generalization bound* (6)*. Then RS satisfies the generalization bound with a constant factor loss in failure probability and approximation factor.*

4 Proof of Lemma 9

Proof. Each column S_i of the subsampled randomized Hadamard transform has the same distribution as σ_iS_i, where σ_i is a random sign. It also has $\langle S_i, S_i\rangle = 1$ for all i and $|\langle S_i, S_j\rangle| \lesssim \frac{\sqrt{\log(1/\delta)}}{\sqrt{r}}$ with probability $1 - \delta$, for any δ and $i \neq j$. See, e.g., [17].

By expanding the following product into a sum, and rearranging terms, we obtain

$$\begin{aligned} a^\top S^\top S(UU^\top S^\top S)^k b = & \sum_{i_0,j_0,i_1,j_1,\cdots,i_k,j_k} a_{i_0} b_{j_k} \sigma_{i_0}\sigma_{i_1}\cdots\sigma_{i_k}\sigma_{j_0}\sigma_{j_1}\cdots\sigma_{j_k} \\ & \cdot \langle S_{i_0}, S_{j_0}\rangle (UU^\top)_{j_0,i_1}\langle S_{i_1}, S_{j_1}\rangle \cdots (UU^\top)_{j_{k-1},i_k}\langle S_{i_k}, S_{j_k}\rangle \\ = & \sum_{i_0,j_k} a_{i_0} b_{j_k}\sigma_{i_0}\sigma_{j_k} \sum_{j_0,i_1,j_1,\cdots,i_k} \sigma_{i_1}\cdots\sigma_{i_k}\sigma_{j_0}\sigma_{j_1}\cdots\sigma_{j_{k-1}} \\ & \cdot \langle S_{i_0}, S_{j_0}\rangle (UU^\top)_{j_0,i_1}\langle S_{i_1}, S_{j_1}\rangle \cdots (UU^\top)_{j_{k-1},i_k}\langle S_{i_k}, S_{j_k}\rangle \\ = & \sum_{i_0,j_k} \sigma_{i_0}\sigma_{j_k} Z_{i_0,j_k} \end{aligned}$$

where Z_{i_0,j_k} is defined to be

$$Z_{i_0,j_k} \quad = \quad a_{i_0} b_{j_k} \sum_{\substack{i_1,\cdots i_k \\ j_0,\cdots j_{k-1}}} \prod_{c=1}^{k} \sigma_{i_c} \prod_{c=0}^{k-1} \sigma_{j_c} \cdot \prod_{c=0}^{k} \langle S_{i_c}, S_{j_c}\rangle \prod_{c=1}^{k} (UU^\top)_{i_{c-1},j_c}.$$

Note that Z_{i_0,j_k} is independent of σ_{i_0} and σ_{j_k}. We observe that in the above expression if $i_0 = j_0, i_1 = j_1, \cdots, i_k = j_k$, then the sum over these indices equals $a^\top(UU^\top)\cdots(UU^\top)b = 0$, since $\langle S_{i_c}, S_{j_c}\rangle = 1$ in this case for all c. Moreover, the sum over all indices conditioned on $i_k = j_k$ is equal to 0. Indeed, in this case, the expression can be factored into the form $\zeta \cdot U^\top b$, for some random variable ζ, but $U^\top b = 0$.

Let W be a matrix with $W_{i,j} = \sigma_i\sigma_j Z_{i,j}$. We need Khintchine's inequality:

▶ **Fact 12** (Khintchine's Inequality). *Let $\sigma_1, \ldots, \sigma_n$ be i.i.d. sign random variables, and let $z_1, \ldots, z_n$ be real numbers. Then there are constants $C, C' > 0$ so that* $\Pr[|\sum_{i=1}^n z_i\sigma_i| \geq Ct\|z\|_2] \leq e^{-C't^2}$.

We note that Khintchine's inequality sometimes refers to bounds on the moment of $|\sum_i z_i\sigma_i|$, though the above inequality follows readily by applying a Markov bound to the high moments.

We apply Fact 12 to each column of W, so that if W_i is the i-th column, we have by a union bound that with probability $1 - 1/\mathrm{poly}(n)$, $\|W_i\|_2 = O(\|Z_i\|_2\sqrt{\log n})$ simultaneously for all columns i. It follows that with the same probability, $\|W\|_F^2 = O(\|Z\|_F^2 \log n)$, that is, $\|W\|_F = O(\|Z\|_F\sqrt{\log n})$. We condition on this event in the remainder.

Thus, it remains to bound $\|Z\|_F$. By squaring Z_{i_0,j_0} and using that $\mathbf{E}[\sigma_i\sigma_j] = 1$ if $i = j$ and 0 otherwise, we have,

$$\mathop{\mathbf{E}}_{\sigma}[Z_{i_0,j_k}^2] = a_{i_0}^2 b_{j_k}^2 \sum_{\substack{i_1,\cdots i_k \\ j_0,\cdots j_{k-1}}} \prod_{c=0}^{k} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k} (UU^\top)_{i_{c-1},j_c}^2. \tag{10}$$

Due to space considerations, we defer to the full version Appendix E the proof that

$$\mathop{\mathbf{E}}_{S}[\|Z\|_F^2] \leq (O(d(\log n)/r) + 1)^{k-1} \cdot (d\|b\|_2^2(\log^2 n)/r^2 + \|b\|_2^2(\log n)/r).$$

Note that we also have the bound:

$$(O(d(\log n)/r) + 1)^{k-1} \leq (e^{O(d(\log n)/r)})^{k-1} \leq e^{O(kd(\log n)/r)} \leq O(1),$$

for any $r = \Omega(kd\log n)$.

Having computed the expectation of $\|Z\|_F^2$, we now would like to show concentration. Consider a specific

$$Z_{i_0,j_k} = a_{i_0} b_{j_k} \sum_{i_k} \sigma_{i_k} \langle S_{i_k}, S_{j_k} \rangle \cdots \sum_{j_1} \sigma_{j_1} (UU^\top)_{j_1,i_2} \sum_{i_1} \sigma_{i_1} \langle S_{i_1}, S_{j_1} \rangle \sum_{j_0} \sigma_{j_0} \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0,i_1}.$$

By Fact 12, for each fixing of i_1, with probability $1 - 1/\mathrm{poly}(n)$, we have

$$\sum_{j_0} \sigma_{j_0} \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0,i_1} = O(\sqrt{\log n})(\sum_{j_0} \langle S_{i_0}, S_{j_0} \rangle^2 (UU^\top)_{j_0,i_1}^2)^{\frac{1}{2}}. \tag{11}$$

Now, we can apply Khintchine's inequality for each fixing of j_1, and combine this with (11). With probability $1 - 1/\mathrm{poly}(n)$, again we have

$$\begin{aligned} &\sum_{i_1} \sigma_{i_1} \langle S_{i_1}, S_{j_1} \rangle \sum_{j_0} \sigma_{j_0} \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0,i_1} \\ = &\sum_{i_1} \sigma_{i_1} \langle S_{i_1}, S_{j_1} \rangle O(\sqrt{\log n})(\sum_{j_0} \langle S_{i_0}, S_{j_0} \rangle^2 (UU^\top)_{j_0,i_1}^2)^{\frac{1}{2}} \\ = &O(\log n)(\sum_{i_1} \langle S_{i_1}, S_{j_1} \rangle^2 \sum_{j_0} \langle S_{i_0}, S_{j_0} \rangle^2 (UU^\top)_{j_0,i_1}^2)^{\frac{1}{2}}. \end{aligned}$$

Thus, we can apply Khintchine's inequality recursively over all the $2k$ indexes $j_0, i_1, j_1, \cdots, j_{k-1}$, i_k, from which it follows that with probability $1 - 1/\mathrm{poly}(n)$, for each such i_0, j_k, we have $Z_{i_0,j_k}^2 = O(\log^k n) \mathop{\mathbf{E}}_S [Z_{i_0,j_k}^2]$, using (10). We thus have with this probability, that $\|Z\|_F^2 = O(\log^k n) \mathop{\mathbf{E}}_S [\|Z\|_F^2]$, completing the proof. ◀

5 Lower bound for ℓ_2 and ℓ_∞ guarantee

We prove a lower bound for the ℓ_2 guarantee, which immediately implies a lower bound for the ℓ_∞ guarantee.

▶ **Definition 13.** Given a matrix $A \in \mathbb{R}^{n \times d}$, vector $b \in \mathbb{R}^n$ and matrix $S \in \mathbb{R}^{r \times n}$, denote $x^* = A^\dagger b$. We say that an algorithm $\mathcal{A}(A, b, S)$ that outputs a vector $x' = (SA)^\dagger Sb$ "succeeds" if the following property holds: $\|x' - x^*\|_2 \lesssim \epsilon \|b\|_2 \cdot \|A^\dagger\|_2 \cdot \|Ax^* - b\|_2$.

▶ **Theorem 14.** *Suppose Π is a distribution over $\mathbb{R}^{m \times n}$ with the property that for any $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, $\Pr_{S \sim \Pi}[\mathcal{A}(A, b, S)$ succeeds $] \geq 19/20$. Then $m \gtrsim \min(n, d/\epsilon^2)$.*

Proof. The proof uses Yao's minimax principle. Let $\mathcal{D}$ be an arbitrary distribution over $\mathbb{R}^{n \times (d+1)}$, then $\mathop{\mathbb{E}}_{(A,b) \sim \mathcal{D}} \mathop{\mathbb{E}}_{S \sim \Pi} [\mathcal{A}(A, b, S)$ succeeds $] \geq 1 - \delta$. Switching the order of probabilistic quantifiers, an averaging argument implies the existence of a fixed matrix $S_0 \in \mathbb{R}^{m \times n}$ such that $\mathop{\mathbb{E}}_{(A,b) \sim \mathcal{D}} [\mathcal{A}(A, b, S_0)$ succeeds $] \geq 1 - \delta$. Thus, we must construct a distribution $\mathcal{D}_{\mathrm{hard}}$ such that $\mathop{\mathbb{E}}_{(A,b) \sim \mathcal{D}_{\mathrm{hard}}} [\mathcal{A}(A, b, S_0)$ succeeds $] \geq 1 - \delta$ cannot hold for any $\Pi_0 \in \mathbb{R}^{m \times n}$ which does not satisfy $m = \Omega(d/\epsilon^2)$. The proof can be split into three parts. First, we prove a useful property. Second, we prove a lower bound for the case $\mathrm{rank}(S) \geq d$. Third, we show why $\mathrm{rank}(S) \geq d$ is necessary.

(I) We show that $[SA, Sb]$ are independent Gaussian, if both $[A, b]$ and S are orthonormal matrices. We can rewrite SA in the following sense,

$$\underbrace{S}_{m\times n}\cdot\underbrace{A}_{n\times d} = \underbrace{S}_{m\times n}\underbrace{R}_{n\times n}\underbrace{R^\top}_{n\times n}\underbrace{A}_{n\times d} \tag{12}$$
$$= S\begin{bmatrix}S^\top & \overline{S}^\top\end{bmatrix}\begin{bmatrix}S\\ \overline{S}\end{bmatrix}A = \begin{bmatrix}I_m & 0\end{bmatrix}\begin{bmatrix}S\\ \overline{S}\end{bmatrix}A = \begin{bmatrix}I_m & 0\end{bmatrix}\underbrace{\widetilde{A}}_{n\times d} = \underbrace{\widetilde{A}_m}_{m\times d}$$

where $\overline{S}$ is the complement of the orthonormal basis S, I_m is a $m \times m$ identity matrix, and $\widetilde{A}_m$ is the left $m \times d$ submatrix of $\widetilde{A}$. Thus, using [13] as long as $m = o(\sqrt{n})$ (because of $n = \Omega(d^3)$) the total variation distance between $[SA, Sb]$ and a random Gaussian matrix is small, i.e.,

$$D_{TV}([SA, Sb], H) \leq 0.01 \tag{13}$$

where each entry of H is i.i.d. Gaussian $\mathcal{N}(0, 1/n)$.

(II) Here we prove the theorem in the case when S has rank $r \geq d$ (we will prove this is necessary in part III. Writing $S = U\Sigma V^\top$ in its SVD, we have

$$\underbrace{S}_{m\times n}A = \underbrace{U}_{m\times r}\underbrace{\Sigma}_{r\times r}\underbrace{V^\top}_{r\times n}RR^\top A = U\Sigma G \tag{14}$$

where $R = \begin{bmatrix}V & \overline{V}\end{bmatrix}$. By a similar argument in Equation (12), as long as $r = o(\sqrt{n})$ we have that G also can be approximated by a Gaussian matrix, where each entry is sampled from i.i.d. $\mathcal{N}(0, 1/n)$. Similarly, $Sb = U\Sigma h$, where h also can be approximated by a Gaussian matrix, where each entry is sampled from i.i.d. $\mathcal{N}(0, 1/n)$.

Since U has linearly independent columns, $(U\Sigma G)^\dagger U\Sigma h = (\Sigma G)^\dagger U^\top U\Sigma h = (\Sigma G)^\dagger \Sigma h$.

The $r \times d$ matrix G has SVD $G = \underbrace{R}_{r\times d}\underbrace{\widetilde{\Sigma}}_{d\times d}\underbrace{T}_{d\times d}$, and applying the pseudo-inverse property again, we have

$$\|(SA)^\dagger Sb\|_2 = \|(\Sigma G)^\dagger \Sigma h\|_2 = \|(\Sigma R\widetilde{\Sigma}T)^\dagger \Sigma h\|_2 = \|T^\dagger(\Sigma R\widetilde{\Sigma})^\dagger \Sigma h\|_2 = \|(\Sigma R\widetilde{\Sigma})^\dagger \Sigma h\|_2$$
$$= \|\widetilde{\Sigma}^\dagger(\Sigma R)^\dagger \Sigma h\|_2,$$

where the the first equality follows by Equation (14), the second equality follows by the SVD of G, the third and fifth equality follow by properties of the pseudo-inverse when T has orthonormal rows and $\widetilde{\Sigma}$ is a diagonal matrix, and the fourth equality follows since $\|T^\dagger\|_2 = 1$ and T is an orthonormal basis.

Because each entry of $G = R\widetilde{\Sigma}T \in \mathbb{R}^{r\times d}$ is sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1)$, using the result of [24] we can give an upper bound for the maximum singular value of G: $\|\widetilde{\Sigma}\| \lesssim \sqrt{\frac{r}{n}}$ with probability at least .99. Thus,

$$\|\widetilde{\Sigma}^\dagger(\Sigma R)^\dagger \Sigma h\|_2 \geq \sigma_{\min}(\widetilde{\Sigma}^\dagger)\cdot\|(\Sigma R)^\dagger \Sigma h\|_2 = \sigma_{\max}^{-1}(\widetilde{\Sigma})\|(\Sigma R)^\dagger \Sigma h\|_2 \gtrsim \sqrt{n/r}\|(\Sigma R)^\dagger \Sigma h\|_2.$$

Because h is a random Gaussian vector which is independent of $(\Sigma R)^\dagger\Sigma$, $\mathbf{E}_h[\|(\Sigma R)^\dagger \Sigma h\|_2^2] = \frac{1}{n}\cdot\|(\Sigma R)^\dagger\Sigma\|_F^2$, where each entry of h is sampled from i.i.d. Gaussian $\mathcal{N}(0, 1/n)$. Then, using the Pythagorean Theorem, $\|(\Sigma R)^\dagger\Sigma\|_F^2 = \|(\Sigma R)^\dagger\Sigma RR^\top\|_F^2 + \|(\Sigma R)^\dagger\Sigma(I - RR^\top)\|_F^2 \geq \|(\Sigma R)^\dagger\Sigma RR^\top\|_F^2 = \|(\Sigma R)^\dagger\Sigma R\|_F^2 = \text{rank}(\Sigma R) = \text{rank}(SA) = d$. Thus, $\|x' - x^*\|_2 \gtrsim \sqrt{d/r} \geq \sqrt{d/m} = \epsilon$.

(III) Now we show that we can assume that $\text{rank}(S) \geq d$.

We sample A, b based on the following distribution $\mathcal{D}_{\text{hard}}$: with probability $1/2$, A, b are sampled from $\mathcal{D}_1$; with probability $1/2$, A, b are sampled from $\mathcal{D}_2$. In distribution $\mathcal{D}_1$, A is a random orthonormal basis and d is always orthogonal to A. In distribution $\mathcal{D}_2$, A is a $d \times d$ identity matrix in the top-d rows and 0s elsewhere, while b is a random unit vector. Then, for any (A, b) sampled from $\mathcal{D}_1$, S needs to work with probability at least $9/10$. Also for any (A, b) sampled from $\mathcal{D}_2$, S needs to work with probability at least $9/10$. The latter two statements follow since overall S succeeds on $\mathcal{D}_{\text{hard}}$ with probability at least $19/20$.

Consider the case where A, b are sampled from distribution $\mathcal{D}_2$. Then $x^* = b$ and $\text{OPT} = 0$. Then consider x' which is the optimal solution to $\min_x \|SAx - Sb\|_2^2$, so $x' = (SA)^\dagger Sb = (S_L)^\dagger S_L b$, where S can be decomposed into two matrices $S_L \in \mathbb{R}^{r \times d}$ and $S_R \in \mathbb{R}^{r \times (n-d)}$, $S = \begin{bmatrix} S_L & S_R \end{bmatrix}$. Plugging x' into the original regression problem, $\|Ax' - b\|_2^2 = \|A(S_L)^\dagger S_L b - b\|_2^2$, which is at most $(1+\epsilon)\,\text{OPT} = 0$. Thus $\text{rank}(S_L)$ is d. Since S_L is a submatrix of S, the rank of S is also d. ◀

References

1. Patrick Amestoy, Iain S. Duff, Jean-Yves L'Excellent, Yves Robert, François-Henry Rouet, and Bora Uçar. On computing inverse entries of a sparse matrix in an out-of-core environment. *SIAM J. Scientific Computing*, 34(4), 2012.
2. Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM J. Scientific Computing*, 32(3):1217–1236, 2010.
3. Jean Bourgain, Sjoerd Dirksen, and Jelani Nelson. Toward a unified theory of sparse dimensionality reduction in euclidean space. In *STOC*, 2015.
4. Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2009.
5. Kenneth L Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.
6. Michael B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *SODA*, pages 278–287, 2016.
7. Michael B Cohen, Cameron Musco, and Christopher Musco. Ridge leverage scores for low-rank approximation. *SODA*, 2017.
8. Michael B Cohen, Jelani Nelson, and David P. Woodruff. Optimal approximate matrix product in terms of stable rank. In *ICALP*, 2016.
9. Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.
10. Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13:3475–3506, 2012.
11. Petros Drineas, Michael W Mahoney, S Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011.
12. François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC'14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.
13. Tiefeng Jiang. How many entries of a typical orthogonal matrix can be approximated by independent normals? *The Annals of Probability*, 34(4):1497–1529, 2006.
14. Jeff Leek. Prediction: the lasso vs. just using the top 10 predictors. In *simplystats, February 23.* `http://simplystatistics.org/2012/02/23/prediction-the-lasso-vs-just-using-the-top-10/`, 2012.

15 Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 127–136, 2013.

16 Song Li, Shaikh S. Ahmed, Gerhard Klimeck, and Eric Darve. Computing entries of the inverse of a sparse matrix using the FIND algorithm. *J. Comput. Physics*, 227(22):9408–9427, 2008.

17 Yichao Lu, Paramveer Dhillon, Dean Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *NIPS*, 2013.

18 Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100. ACM, 2013.

19 Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 117–126. IEEE, 2013.

20 Jelani Nelson and Huy L. Nguyên. Lower bounds for oblivious subspace embeddings. In *ICALP*, pages 883–894, 2014.

21 Dimitris Papailiopoulos, Anastasios Kyrillidis, and Christos Boutsidis. Provable deterministic leverage score sampling. In *KDD*, pages 997–1006. ACM, 2014.

22 Ryan Rifkin, Gene Yeo, and Tomaso Poggio. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.

23 Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

24 Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

25 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19-22, 2012*, pages 887–898, 2012.

26 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

27 Peng Zhang and Jing Peng. Svm vs regularized least squares classification. In *ICPR (1)*, pages 176–179, 2004.

28 Anastasios Zouzias and Nikolaos M. Freris. Randomized extended kaczmarz for solving least squares. *SIAM J. Matrix Analysis Applications*, 34(2):773–793, 2013.

Embeddings of Schatten Norms with Applications to Data Streams*

Yi Li[1] **and David P. Woodruff**[2]

1 **Division of Mathematics, School of Physical & Mathematical Sciences, Nanyang Technological University, Singapore**
yili@ntu.edu.sg

2 **IBM Almaden Research Center, San Jose, CA, USA**
dpwoodru@us.ibm.com

Abstract

Given an $n \times d$ matrix A, its Schatten-p norm, $p \geq 1$, is defined as $\|A\|_p = \left(\sum_{i=1}^{\mathrm{rank}(A)} \sigma_i(A)^p\right)^{1/p}$, where $\sigma_i(A)$ is the i-th largest singular value of A. These norms have been studied in functional analysis in the context of non-commutative ℓ_p-spaces, and recently in data stream and linear sketching models of computation. Basic questions on the relations between these norms, such as their embeddability, are still open. Specifically, given a set of matrices $A^1, \ldots, A^{\mathrm{poly}(nd)} \in \mathbb{R}^{n \times d}$, suppose we want to construct a linear map L such that $L(A^i) \in \mathbb{R}^{n' \times d'}$ for each i, where $n' \leq n$ and $d' \leq d$, and further, $\|A^i\|_p \leq \|L(A^i)\|_q \leq D_{p,q}\|A^i\|_p$ for a given approximation factor $D_{p,q}$ and real number $q \geq 1$. Then how large do n' and d' need to be as a function of $D_{p,q}$?

We nearly resolve this question for every $p, q \geq 1$, for the case where $L(A^i)$ can be expressed as $R \cdot A^i \cdot S$, where R and S are arbitrary matrices that are allowed to depend on $A^1, \ldots, A^t$, that is, $L(A^i)$ can be implemented by left and right matrix multiplication. Namely, for every $p, q \geq 1$, we provide nearly matching upper and lower bounds on the size of n' and d' as a function of $D_{p,q}$. Importantly, our upper bounds are *oblivious*, meaning that R and S do not depend on the A^i, while our lower bounds hold even if R and S depend on the A^i. As an application of our upper bounds, we answer a recent open question of Blasiok et al. about space-approximation trade-offs for the Schatten 1-norm, showing in a data stream it is possible to estimate the Schatten-1 norm up to a factor of $D \geq 1$ using $\tilde{O}(\min(n, d)^2/D^4)$ space.

1998 ACM Subject Classification G. Mathematics of Computing

Keywords and phrases data stream algorithms, embeddings, matrix norms, sketching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.60

1 Introduction

Given an $n \times d$ matrix A, its Schatten-p norm, $p \geq 1$, is defined as $\|A\|_p = \left(\sum_{i=1}^{r(A)} \sigma_i(A)^p\right)^{\frac{1}{p}}$, where $r(A)$ is the rank of A and $\sigma_i(A)$ is the i-th largest singular value of A, i.e., the square root of the i-th largest eigenvalue of $A^T A$. The Schatten-1 norm is the nuclear norm or trace norm, the Schatten-2 norm is the Frobenius norm, and the Schatten ∞-norm, defined as the limit of the Schatten-p norm when $p \to \infty$, is the operator norm. The Schatten 1-norm has applications in non-convex optimization [5], while Schatten-2 and Schatten-∞ norms are useful in geometry and linear algebra, see, e.g., [22]. Schatten-p norms for large p also provide approximations to the Schatten-∞ norm.

* A full version of the paper is available at https://arxiv.org/abs/1702.05626.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 60; pp. 60:1–60:14

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

The Schatten norms appear to be significantly harder to compute or approximate than the vector ℓ_p-norms in various models of computation, and understanding the complexity of estimating them has led to new algorithmic ideas and lower bound techniques. The main difficulty is that we do not directly have access to the spectrum of A, and naïvely it is costly in space and time to extract useful information about it. A line of work has focused on understanding the complexity of estimating such norms in the data stream model with 1-pass over the stream [13] as well as with multiple passes [4], the sketching model [2, 12, 14], statistical models [9], as well as the general RAM model [17, 19]. Dimensionality reduction in these norms also has applications in quantum computing [8, 21]. It has also been asked in places if the Schatten-1 norm admits non-trivial nearest neighbor search data structures [1].

Our Results. In this paper we study the embeddability of the Schatten-p norm into the Schatten-q norm for linear maps implementable by matrix multiplication. More concretely, we first ask for the following form of embeddability: given n and t (where $t = \Omega(\log n)$), what is the smallest value of $D_{p,q}$, which we call the *distortion*, such that there exists a distribution $\mathcal{R}$ on $\mathbb{R}^{t\times n}$ satisfying, for any given $n \times d$ matrix A,

$$\Pr_{R\sim\mathcal{R}}\left\{\|A\|_p \le \|RA\|_q \le D_{p,q}\|A\|_p\right\} \ge 1 - \exp(-ct)?$$

Here $c > 0$ is an absolute constant. We can assume, w.l.o.g., that $n = d$ because we can first apply a so-called *subspace embedding* matrix (see, e.g., [22] for a survey) to the left or to the right of A to preserve each of its singular values up to a constant factor. We shall show that $D_{p,q} \gtrsim \hat{D}_{p,q}$, where

$$\hat{D}_{p,q} = \begin{cases} n^{\frac{1}{p}-\frac{1}{2}}/t^{\frac{1}{q}-\frac{1}{2}}, & 1 \le p \le q \le 2; \\ n^{\frac{1}{p}-\frac{1}{2}}, & 1 \le p \le 2 \le q; \\ \max\{(n/t)^{\frac{1}{2}-\frac{1}{p}}, t^{\frac{1}{p}-\frac{1}{q}}\}, & 2 \le p \le q; \\ n^{\frac{1}{2}-\frac{1}{p}}, & 1 \le q \le 2 \le p; \\ n^{\frac{1}{2}-\frac{1}{p}}/t^{\frac{1}{2}-\frac{1}{q}}, & 2 \le q \le p; \\ \max\{(n/t)^{\frac{1}{p}-\frac{1}{2}}, (t/\ln t)^{\frac{1}{q}-\frac{1}{p}}\}, & 1 \le q \le p \le 2, \end{cases} \tag{1}$$

and the notation $f \gtrsim g$ means $f \ge g/C$ for some constant $C > 0$. The constant C in the $\gtrsim$ notation above depends on p and q only. This distortion is asymptotically tight, up to logarithmic factors, as we also construct a distribution $\mathcal{R}$ on t-by-n matrices for which for any $n \times d$ matrix A,

$$\Pr_{R\sim\mathcal{R}}\left\{\|A\|_p \le \|RA\|_q \le \tilde{D}_{p,q}\left(\log\frac{n}{t}\right)\|A\|_p\right\} \ge 1 - \exp\left(-ct\right),$$

where $\tilde{D}_{p,q}$ differs from $D_{p,q}$ by a constant or a factor of $\log t$. Specifically,

$$\tilde{D}_{p,q} \lesssim \begin{cases} \max\{(n/t)^{\frac{1}{p}-\frac{1}{2}}, t^{\frac{1}{q}-\frac{1}{p}}\}, & 1 \le q \le p \le 2; \\ \hat{D}_{p,q}, & \text{otherwise}, \end{cases} \tag{2}$$

where $\hat{D}_{p,q}$ is given in (1). Replacing t with $t/(\ln(n/t))$, we arrive at a matching failure probability and distortion, while using a logarithmic factor more number of rows in R. Namely, we construct a distribution $\mathcal{R}$ on matrices with $t\ln(n/t)$ rows for which

$$\Pr_{R\sim\mathcal{R}}\left\{\|A\|_p \le \|RA\|_q \le \tilde{D}_{p,q}\|A\|_p\right\} \ge 1 - \exp\left(-ct\right).$$

We can also sketch RA on the right by a subspace embedding matrix S with $\Theta(t)$ rows, which yields

$$\Pr_{R,S}\left\{\|A\|_p \leq \left\|RAS^T\right\|_q \leq \tilde{D}_{p,q}\|A\|_p\right\} \geq 1 - \exp\left(-ct\right).$$

We show that this two-sided sketch is asympotically optimal for two-sided sketches in its product of number of rows of R and number of columns of S, up to logarithmic factors. Formally, we next ask: what is the smallest value of $D_{p,q}$ for which there exists a distribution $\mathcal{G}_1$ on $\mathbb{R}^{r\times n}$ and a distribution $\mathcal{G}_2$ on $\mathbb{R}^{n\times s}$ satisfying

$$\Pr_{R\sim\mathcal{G}_1, S\sim\mathcal{G}_2}\left\{\|A\|_p \leq \|RAS\|_q \leq D_{p,q}\|A\|_p\right\} \geq 1 - \exp(-c\min\{r,s\})?$$

Again we can assume, w.l.o.g, that $r = s$, because otherwise we can compose R or S with a subspace embedding to preserve all singular values up to a constant factor[1]. Henceforth for the two-sided problem, we assume that $\mathcal{G}_1$ and $\mathcal{G}_2$ are distributions on $\mathbb{R}^{t\times n}$. We also prove a matching lower bound that $D_{p,q} \gtrsim \hat{D}_{p,q}$ except in the case when $1 \leq q \leq p \leq 2$, where we instead obtain a matching lower bound up to logarithmic factors, namely, $D_{p,q} \gtrsim \max\{(n/t)^{\frac{1}{p}-\frac{1}{2}}/\log^{\frac{3}{2}} t, (t/\ln t)^{\frac{1}{q}-\frac{1}{p}}\}$.

In the important case when $p = q = 1$, our results show a space-approximation tradeoff for estimating the Schatten 1-norm (or trace norm) in a data stream, answering a question posed by Blasiok et al. [3]. This application crucially uses that R and S are oblivious to A, i.e., they can be sampled and succinctly stored without looking at A. Specifically, when each entry of A fits in a word of $O(\log n)$ bits, we can choose R and S to be Gaussian random matrices with entries truncated to $O(\log n)$ bits and with entries drawn from a family of random variables with bounded independence. For time-efficiency purposes, R and S can also be chosen to be Fast Johnson Lindenstrauss Transforms or sparse embedding matrices [6, 16, 18], though they will have larger dimension, especially to satisfy the exponential probability of failure in the problem statement (and even with constant failure probability, the dimension will be slightly larger; see [22] for a survey).

Choosing R and S to be Gaussian matrices, our result provides a data stream algorithm using $(n^2/D^4)\,\mathrm{polylog}(n)$ bits of memory, and achieving approximation factor D (taking $t = n/D^2$). While $\|A\|_2$, the Frobenius norm of A, provides a $\sqrt{n}$-approximation to $\|A\|_1$ and can be approximated up to a constant factor in a data stream using $O(1)$ words of space, if we want an algorithm achieving a better approximation factor then all that was known was an algorithm requiring $O(n^2)$ words of space, namely, the trivial algorithm of storing A exactly and achieving $D = 1$. It was asked in [3] if there is a smooth trade-off between the case when $D = 1$ and $D = \sqrt{n}$; our $(n^2/D^4)\,\mathrm{polylog}(n)$ space algorithm provides the first such trade-off, and is optimal at the two extremes. Our results are the first of their kind for large approximation factors $D \gg 1$ for estimating the Schatten-p norms in a data stream.

Finally, while in our upper bounds R and S are chosen obliviously to A, for our lower bounds we would like to rule out those R and S which are even allowed to depend on A. Clearly, if there is only a single matrix A, this question is ill-posed as one can just choose R and S to have a single row and column so that $\|RAS\|_q = \|A\|_p$. Instead, we ask the question analogous to the Johnson-Lindenstrauss transform (see e.g., [10]): given

[1] That is, if $r \leq s$, we can choose a subspace embedding matrix H of dimension $n \times \Theta(r)$ such that $\|RASH\|_q = \Theta(\|RAS\|_q)$ with probability $\geq 1 - \exp(-s)$, and then pad R with zero rows so that R has the same number of rows as columns of S, increasing the number of rows of R by at most a constant factor.

$A^1, \ldots, A^{\mathrm{poly}(n)}$, can we construct an R with t rows and an S with t columns for which $\|A^i\|_p \leq \|RA^iS\|_q \leq D_{p,q}\|A^i\|_p$ for all i? We show that our lower bound on the trade-off between $D_{p,q}$ and t given by (1) continues to hold even in this setting.

Our Techniques. We shall focus on the case $p = q$ in this description of our technical overview. For our upper bounds, a natural idea is to take R to be a (normalized) Gaussian random matrix, and the analysis of the quantity $\|RA\|_p$, when $p \geq 2$, follows fairly directly from the so-called non-commutative Khintchine inequality as follows.

▶ **Lemma 1** (Non-commutative Khintchine Inequality [15]). *Suppose that $C_1, \ldots, C_n$ are (deterministic) matrices of the same dimension and $g_1, \ldots, g_n$ are independent $N(0,1)$ variables. It holds that*

$$\mathop{\mathbb{E}}_{g_1,\ldots,g_n} \left\| \sum_i g_i C_i \right\|_p \simeq \max\left\{ \left\| \left(\sum_i C_i C_i^T \right)^{\frac{1}{2}} \right\|_p, \left\| \left(\sum_i C_i^T C_i \right)^{\frac{1}{2}} \right\|_p \right\}, \quad p \geq 2.$$

In order to estimate $\|RA\|_p$, we can write

$$RA = \sum_{i,j} r_{ij}(e_i e_j^T A) =: \sum_{i,j} r_{ij} C_{ij}$$

and it is straightforward to compute that

$$\sum_{i,j} C_{ij} C_{ij}^T = \mathrm{tr}(AA^T) I_t = \|A\|_F^2 I_t, \quad \sum_{i,j} C_{ij}^T C_{ij} = t \cdot A^T A.$$

It follows from the non-commutative Khintchine inequality that (recall that R is a normalized Gaussian matrix with $N(0, 1/t)$ entries)

$$\mathbb{E}\,\|RA\|_p \simeq \max\left\{ t^{\frac{1}{2}-\frac{1}{p}} \|A\|_F, \|A\|_p \right\}, \quad p \geq 2.$$

Using a concentration inequality for Lipschitz functions on Gaussian space, one can show that $\|RA\|_p$ is concentrated around $\mathbb{E}\,\|RA\|_p$, and using standard the standard relationship between $\|A\|_F$ and $\|A\|_p$ then completes the argument.

When $p < 2$, the non-commutative Khintchine inequality gives a much less tractable characterization, so we need to analyze $\|RA\|_p$ in a different manner, which is potentially of independent interest. Our analysis also works for non-Gaussian matrices R whenever R satisfies certain properties, which, for instance, are satisfied by a Fast Johnson-Lindenstrauss Transform.

Upper bound. We give an overview of our upper bound now, focusing on the one-sided case, since the two-sided case follows by simply right-multiplying by a generic subspace embedding S. Here we focus on the case in which R is an $r \times n$ Gaussian matrix, where $r = t \cdot \mathrm{polylog}(n)$. By rotational invariance of Gaussian matrices, and for the purposes of computing $\|AR\|_p$, we can assume that A is diagonal. Let A_1 be the restriction of A to its top $\Theta(t \log n)$ singular values. Since R is a Gaussian matrix with at least $t \log n$ rows, it is well-known that R is also a subspace embedding on A_1 (see, e.g., [20, Corollary 5.35]), namely, $\sigma_i(RA_1) \simeq \sigma_i(A_1)$ for all i, and thus $\|RA_1\|_p \simeq \|A_1\|_p = \Omega(\|A\|_p)$ when $\|A_1\|_p = \Omega(\|A\|_p)$.

If it does not hold that $\|A_1\|_p = \Omega(\|A\|_p)$, then the singular values of A are "heavy-tailed", and we show how to find a $\sigma_i(A)$ with $i < \Theta(t \log n)$ for which $\sigma_i^2(A)$ is relatively small compared to $\sigma_i^2(A) + \sigma_{i+1}^2(A) + \cdots + \sigma_n^2(A)$. More specifically, let A_2 be the restriction

of A to $\sigma_i(A), \ldots, \sigma_n(A)$. Then we have that $\|A_2\|_{op} \lesssim \|A_2\|_F/\sqrt{t}$. Since for a Gaussian matrix R it holds that $\|RA_2\|_{op} \lesssim \|A_2\|_{op} + \|A\|_F/\sqrt{r}$ (see Proposition 3), we thus have that $\|RA_2\|_{op} \lesssim \|A_2\|_F/\sqrt{t}$. On the other hand, $\|RA_2\|_F \simeq \|A_2\|_F$. This implies there exist $\Omega(t)$ singular values of RA_2 that are $\Omega(\|A_2\|_F/\sqrt{t})$, which yields that $\|RA_2\|_p \gtrsim \|A_2\|_p = \Omega(\|A\|_p)$. Therefore we have established the lower bound that $\|RA\|_p \geq \max\{\|RA_1\|_p, \|RA_2\|_p\}$ in terms of $\|A\|_p$.

To upper bound $\|RA\|_p$ in terms of $\|A\|_p$, note that $\|RA\|_p \leq \|RA_1\|_p + \|RA_2\|_p$ by the triangle inequality, where A_1, A_2 are as above. Again it follows from the subspace embedding property of R that $\|RA_1\|_p \lesssim \|A_1\|_p \leq \|A\|_p$. Regarding $\|RA_2\|_p$, we relate its Schatten-p norm to its Frobenius norm and use the fact that $\|RA_2\|_F \simeq \|A_2\|_F$. This gives an upper bound of $\|RA_2\|_p$ in terms of $\|A_2\|_p$, and using that $\|A_2\|_p \leq \|A\|_p$, it gives an upper bound in terms of $\|A\|_p$. This is sufficient to obtain an overall upper bound on $\|RA\|_p$.

Lower bound. Now we give an overview of our lower bounds for some specific cases. First consider one-sided sketches. We choose our hard distribution as follows: we choose an $n \times (10t)$ Gaussian matrix G padded with 0s to become an $n \times n$ matrix. For a sketch matrix R containing t rows, by rotational invariance of Gaussian matrices, $\|RG\|_p$ is identically distributed to $\|\Sigma_R G'\|_p$, where Σ_R is the $t \times t$ diagonal matrix consisting of the singular values of R, and where G' is a $t \times (10t)$ Gaussian matrix. It is a classical result that all singular values of G' are $\Theta(\sqrt{t})$ and thus $\|RG\|_p \simeq \sqrt{t}\|R\|_p$. This implies that

$$\sqrt{n}t^{\frac{1}{2}-\frac{1}{p}} \lesssim \sqrt{t}\|R\|_p \lesssim D_{p,p}\sqrt{n}t^{\frac{1}{2}-\frac{1}{p}}, \tag{3}$$

since all non-zero singular values of G are $\Theta(\sqrt{n})$. On the other hand, applying R to the $n \times n$ identity matrix gives that

$$n^{\frac{1}{p}} \leq \|R\|_p \leq D_{p,p}n^{\frac{1}{p}}. \tag{4}$$

Combining (3) and (4) gives that $D_{p,p} \geq \max\{(n/t)^{1/2-1/p}, (n/t)^{1/p-1/2}\}$.

For the two-sided sketch, we change the hard distribution to (i) $n \times n$ Gaussian random matrix F and (ii) the distribution of GH^T, where G and H are $n \times \Theta(t)$ Gaussian random matrices. The proof then relies on the analysis for $\left\|RFS^T\right\|_p$ and $\left\|RGH^TS^T\right\|_p$. When $p \geq 2$, non-commutative Khintchine inequality gives immediately that

$$\left\|RGH^TS^T\right\|_p \simeq \sqrt{t}\left\|RFS^T\right\|_p \simeq \sqrt{t}\max\{\|R\|_p\|S\|_{op}, \|R\|_{op}\|S\|_p\}, \quad p \geq 2. \tag{5}$$

When $p < 2$, a different approach is followed. We divide the singular values of R and S into bands, where each band contains singular values within a factor of 2 from each other. We shall consider the first $\Theta(\log t)$ bands only because the remaining singular values are $1/\operatorname{poly}(t)$ and negligible. Now, if all singular values of R' and S' are within a factor of 2 from each other, then $\left\|R'F(S')^T\right\|_p \simeq \|R'\|_{op}\|S'\|_{op}\|F\|_p$ and $\left\|R'GH^T(S')^T\right\|_p \simeq \|R'\|_{op}\|S'\|_{op}\left\|GH^T\right\|_p$. It is not difficult to see that

$$\|GH^T\|_p \simeq \sqrt{t}\|F\|_p \tag{6}$$

Since R' and S' consist of one of the $\Theta(\log t)$ bands of R and S, respectively, it follows that

$$\left\|RGH^TS^T\right\|_p \simeq \sqrt{t}/\operatorname{polylog}(t) \cdot \left\|RFS^T\right\|_p, \quad p < 2. \tag{7}$$

A lower bound of $D_{p,p}$ then follows from combining (6), (5) (or (7)) with

$$\|F\|_p \leq \left\|RFS^T\right\|_p \leq D_{p,p}\|F\|_p, \quad \text{and} \quad \left\|GH^T\right\|_p \leq \left\|RGH^TS^T\right\|_p \leq D_{p,p}\left\|GH^T\right\|_p.$$

To strengthen the lower bound for the sketches that even depend on the input matrix, we follow the approach in [10]. We first work with random hard instances, and then sample input matrices $A^1, \ldots, A^{\mathrm{poly}(n)}$ from the hard distribution, and apply a net argument on sketching matrices R and S to obtain a deterministic statement, which states that for any fixed R and S such that the distortion guarantee is satisfied with all samples $A^1, \ldots, A^{\mathrm{poly}(n)}$, the distortion lower bound remains to hold.

2 Preliminaries

Notation. Throughout the paper, we use $f \lesssim g$ to denote $f \leq Cg$ for some constant C, $f \gtrsim g$ to denote $f \geq Cg$ for some constant C, and $f \simeq g$ to denote $C_1 g \leq f \leq C_2 g$ for some constants C_1 and C_2.

Bands of Singular Values. Given a matrix A, we split the singular values of A, $\sigma_1(A) \geq \sigma_2(A) \geq \cdots$, into bands such that the singular values in each band are within a factor of 2 from each other. Formally, define the i-th singular value band of A to be

$$\mathcal{B}_i(A) = \left\{ k : \frac{\|A\|_{op}}{2^{i+1}} < \sigma_k(A) \leq \frac{\|A\|_{op}}{2^i} \right\}, \quad i \geq 0,$$

and let $N_i(A) = |\mathcal{B}_i(A)|$, the cardinality of the i-th band.

Extreme Singular Values of Gaussian Matrices. We shall repeatedly use the following results on Gaussian matrices.

▶ **Proposition 2** ([20, Corollary 5.35]). *Let G be an $r \times n$ ($r < n$) Gaussian random matrix of i.i.d. entries $N(0,1)$. With probability at least $1 - 2\exp(-u^2/2)$, it holds that $\sqrt{n} - \sqrt{r} - u \leq s_{\min}(G) \leq s_{\max}(G) \leq \sqrt{n} + \sqrt{r} + u$.*

Combining [11, Corollary 3.21] and the concentration bound in Gauss space [20, Proposition 5.34], we also have

▶ **Proposition 3.** *Let A be a deterministic $n \times n$ matrix and G be an $r \times n$ ($r < n$) Gaussian random matrix of i.i.d. entries $N(0,1)$. Then for any K, it holds that $\|GA\|_{op} \leq K(\|A\|_{op}\sqrt{r} + \|A\|_F)$ with probability at least $1 - \exp(-c\sqrt{K}r)$, where $c > 0$ is an absolute constant.*

Nets on Matrices. The following fact concerns nets of matrices and was used in [10]. We shall use it in our lower bound arguments.

▶ **Proposition 4** ([10, Lemma 2]). *There exists a net $\mathcal{R} \subset \bigcup_{t=1}^{t_0} \mathbb{R}^{t \times n}$ of size $\exp(O(t_0 n \ln(Dn/\eta))$ such that for any $R \in \mathbb{R}^{t \times n}$ ($1 \leq t \leq t_0$) with column norms in $[1, D]$, we can find $R' \in \mathcal{R}$ such that $\|R - R'\|_{op} \leq \eta$.*

3 Lower Bounds

In this section we show the full proof of the $(n/t)^{1/2-1/p}$ lower bound for one-sided sketches (Theorem 5) and the $(n/t)^{1/p-1/2}/\log^{3/2} t$ bound for two-sided sketches (Theorem 6), which demonstrates our techniques. Other cases can be found in the full version.

▶ **Theorem 5** (One-sided sketch). *Let $p > 2$ and $p > q$. There exist a set $T \subset \mathbb{R}^{n \times n}$ with $|T| = \mathrm{poly}(n)$ and an absolute constant $c \in (0,1)$ such that, if it holds for some matrix $R \in \mathbb{R}^{t \times n}$ with $t \leq cn$ and for all $A \in T$ that $\|A\|_p \leq \|RA\|_q \leq D_{p,q}\|A\|_p$, then it must hold that $D_{p,q} \gtrsim (n/t)^{\frac{1}{2}-\frac{1}{p}}$.*

Instead of proving this theorem, we prove the following rephrased version.

▶ **Theorem 5'** (rephrased). *Let $p > 2$ and $p > q$. There exists an absolute constant D_0 and a set $T \subset \mathbb{R}^{n\times n}$ with $|T| = O(n\ln(Dn))$ such that, if $D \geq D_0$ and it holds for some matrix $R \in \mathbb{R}^{t\times n}$ and for all $A \in T$ that*

$$\|A\|_p \leq \|RA\|_q \leq D^{\frac{1}{2}-\frac{1}{p}}\|A\|_p, \tag{8}$$

then it must hold that $t \gtrsim n/D$.

Proof. Let $r = n/(\rho^2 D)$ and $t_0 = \theta r$ for some constants $\rho > 1$ and $\theta \in (0,1)$ to be determined. We shall show that if $t \leq t_0$, it will not happen that R satisfies (8) for all A in a carefully chosen set T.

Let $\mathcal{D}$ be the distribution of Gaussian random matrices of dimension $n \times r$ with i.i.d. entries $N(0, 1/r)$. Let $R = U\Sigma V^T$ be the singular value decomposition of R and $A \sim \mathcal{D}$. Then by rotational invariance of the Schatten norm and Gaussian random matrices, we know that $\|RA\|_q$ is identically distributed to $\|\Sigma A\|_q = \|B^T\Sigma'\|_q$, where Σ' is the left $t \times t$ block of Σ and B is formed by the first t rows of A.

It follows from Proposition 2 that with probability $\geq 1 - \exp(-c_1c_2r)$, $s_{\max}(B) \leq 1 + 2c_1\sqrt{t/r} \leq 1 + 2\sqrt{\theta}c_1$, and thus

$$\|B^T\Sigma'\|_q \leq s_{\max}(B)\|\Sigma'\|_q \leq (1+2\sqrt{\theta}c_1)\|\Sigma'\|_q = (1+\sqrt{\theta}c_1)\|R\|_q \leq (1+2\sqrt{\theta}c_1)D^{\frac{1}{2}-\frac{1}{p}}n^{\frac{1}{p}},$$

that is, with probability $\geq 1 - \exp(-c_1c_2r)$,

$$\|RA\|_q \leq (1 + 2\sqrt{\theta}c_1)D^{\frac{1}{2}-\frac{1}{p}}n^{\frac{1}{p}}.$$

On the other hand, with probability $\geq 1 - \exp(-c_1c_2r)$, all singular values of A are at least $\sqrt{n/r} - 2c_1 = \rho\sqrt{D} - 2c_1 \geq (1-\epsilon)\rho\sqrt{D}$ if we choose $D_0 \geq 4c_1^2/\epsilon^2$. Then

$$\|RA\|_q \geq \|A\|_p \geq (1-\epsilon)sr^{\frac{1}{p}}\sqrt{D} = (1-\epsilon)\rho^{1-\frac{2}{p}}n^{\frac{1}{p}}D^{\frac{1}{2}-\frac{1}{p}}.$$

Also, with probability $\geq 1 - \exp(-c_1c_2r)$, all singular values of A are at most $\sqrt{n/r} + 2c_1 = \rho\sqrt{D} + 2c_1 \leq (1+\epsilon)\rho\sqrt{D}$ and thus

$$\|A\|_p \leq r^{\frac{1}{p}}(1+\epsilon)s\sqrt{D} = (1+\epsilon)\rho^{1-\frac{2}{p}}n^{\frac{1}{p}}D^{\frac{1}{2}-\frac{1}{p}}.$$

This motivates the following definitions of constraints for $R \in \mathbb{R}^{t\times n}$ and $A \in \mathbb{R}^{n\times n}$:

$$\mathrm{P}_1(R,A) : \|RA\|_q \leq (1+2\sqrt{\theta}c_1)D^{\frac{1}{2}-\frac{1}{p}}n^{\frac{1}{p}} \quad \mathrm{P}_2(R,A) : \|RA\|_q \geq (1-\epsilon)\rho^{1-\frac{2}{p}}n^{\frac{1}{p}}D^{\frac{1}{2}-\frac{1}{p}}$$
$$\mathrm{P}_3(A) : \|A\|_p \leq (1+\epsilon)\rho^{1-\frac{2}{p}}n^{\frac{1}{p}}D^{\frac{1}{2}-\frac{1}{p}}.$$

Now, for m samples $A_1, \ldots, A_m$ drawn from $\mathcal{D}$, it holds for any fixed R that

$$\Pr_{A_1,\ldots,A_m}\{\exists i \text{ s.t. } \mathrm{P}_1(R,A) \text{ and } \mathrm{P}_2(R,A) \text{ and } \mathrm{P}_3(A) \text{ hold}\} \geq 1 - e^{-c_1c_2mr}. \tag{9}$$

Since $1 \leq \|Ge_ie_i^T\|_q \leq D$ and $\|Ge_ie_i^T\|_q = \|R_i\|_2$, we can restrict the matrix R to matrices with column norm in $[1, D]$. Thus we can find a net $\mathcal{R} \subset \bigcup_{t=1}^{t_0}\mathbb{R}^{t\times n}$ of size $\exp(O(t_0n\ln(Dn/\eta))$ such that for any R with column norms in $[1, D]$, we can find $R' \in \mathcal{R}$ such that $\|R - R'\|_{op} \leq \eta$.

Now it follows from (9) that

$$\Pr_{A_1,\ldots,A_m}\{\forall R \in \mathcal{R}, \exists i,\ \mathrm{P}_1(R,A) \text{ and } \mathrm{P}_2(R,A) \text{ and } \mathrm{P}_3(R,A) \text{ hold}\}$$
$$\geq 1 - \exp\left(O\left(t_0n\ln\frac{Dn}{\eta}\right)\right)\exp\left(-\frac{c_1c_2}{D}mn\right) > 0,$$

provided that $m = \Theta(n \ln(Dn))$. Fix $A_1, \ldots, A_m$ such that for each $R \in \mathcal{R}$ there exists an i such that $\mathrm{P}_1(R, A_i)$ and $\mathrm{P}_2(R, A_i)$ and $\mathrm{P}_3(A_i)$ all hold.

Take $T = \{I_n, e_1 e_1^T, \ldots, e_n e_n^T, A_1, \ldots, A_m\}$. We know that if R satisfies (8) for all $A \in T$, then there exists R' such that $\|R' - R\|_{op} \leq \eta$, and there exists $1 \leq i \leq m$ such that $\mathrm{P}_1(R', A_i)$, $\mathrm{P}_2(R', A_i)$ and $\mathrm{P}_3(A_i)$ all hold. It follows that

$$\begin{aligned}\|RA_i\|_q \leq \|R'A_i\|_q + \|(R - R')A_i\|_q &\leq \|R'A_i\|_q + \|R - R'\|_{op}\|A_i\|_p \\ &\leq \left(1 + 2\sqrt{\theta}c_1 + (1+\epsilon)\rho^{1-\frac{2}{p}}\eta\right) D^{\frac{1}{2}-\frac{1}{p}} n^{\frac{1}{p}}\end{aligned}$$

and

$$\begin{aligned}\|RA_i\|_q \geq \|RA_i\|_q - \|(R - R')A_i\|_q &\geq \|R'A_i\|_q - \|R - R'\|_{op}\|A_i\|_p \\ &\geq \left((1-\epsilon) - (1+\epsilon)\eta\right) \rho^{1-\frac{2}{p}} D^{\frac{1}{2}-\frac{1}{p}} n^{\frac{1}{p}}\end{aligned}$$

We meet a contradiction when θ, ϵ and η are all sufficiently small and ρ is sufficiently large, for instance, when $\eta = \Theta(\epsilon)$, $\theta = \Theta(\epsilon^2/c_2^2)$ and $\rho = \Theta(1 + p\epsilon/(p-2))$. ◀

▶ **Theorem 6** (Two-sided sketch)**.** *Let $p < 2$. There exist a set $T \subset \mathbb{R}^{n \times n}$ with $|T| = \mathrm{poly}(n)$ and an absolute constant $c \in (0,1)$ such that, if it holds for some matrices $R, S \in \mathbb{R}^{t \times n}$ with $t \leq cn$ and for all $A \in T$ that $\|A\|_p \leq \left\|RAS^T\right\|_q \leq D_{p,q}\|A\|_p$, it must hold that $D_{p,q} \gtrsim (n/t)^{\frac{1}{p}-\frac{1}{2}} / \log^{\frac{3}{2}} t$.*

Instead of proving this theorem, we prove the following rephrased version.

▶ **Theorem 6'** (rephrased)**.** *Let $p < 2$, $p > q$ and $D \geq D_0$ for some an absolute constant D_0. There exists a set $T \subset \mathbb{R}^{n \times n}$ with $|T| = O(n \ln(Dn))$ such that it holds for some matrices $R, S \in \mathbb{R}^{t \times n}$ and for all $A \in T$ that*

$$\|A\|_p \leq \left\|RAS^T\right\|_q \leq D^{\frac{1}{p}-\frac{1}{2}}\|A\|_p, \tag{10}$$

then it must hold that $t \gtrsim n/(D \log^{3p/(2-p)} t)$.

We need two auxiliary lemmata, whose proofs are omitted owing to space limitations.

▶ **Lemma 7.** *Let A and B be deterministic $n \times n$ matrices and G be a Gaussian random matrix of i.i.d. $N(0,1)$ entries. It holds with probability $1 - O(1)$ that*

$$\|AGB\|_p \lesssim (\log^{\frac{5}{2}} n)(\log\log n)\|A\|_{op}\|B\|_{op}E_p(A,B),$$

where

$$E_p(A,B) = \max_{0 \leq i,j \leq 3\log n} \frac{1}{2^{i+j}} \cdot \min\left\{N_i(A), N_j(B)\right\}^{\frac{1}{p}} \cdot \max\left\{\sqrt{N_i(A)}, \sqrt{N_j(B)}\right\}. \tag{11}$$

▶ **Lemma 8.** *Let A and B be deterministic $n \times N$ matrices and G, H be $N \times r$ Gaussian random matrices of i.i.d. $N(0,1)$ entries. Suppose that $n \leq cr$ for some absolute constant $c \in (0,1)$. It holds with probability $1 - O(1)$ that $\|AGH^TB^T\|_p \gtrsim \sqrt{r}\|A\|_{op}\|B\|_{op}E_p(A,B)$, where $E_p(A,B)$ is as defined in (11).*

Now we are ready to show Theorem 6'.

Proof of Theorem 6'. Without loss of generality, we can assume that the maximum column norm of R and that of S are the same; otherwise we can rescale R and S.

Let $r = n/(\rho^2 D)$ and $t_0 = \theta r$ for some $\rho = \Theta(\log^{3p/(2-p)} t)$ and $\theta \in (0,1)$ to be determined. We shall show that if $t \le t_0$, it will not happen that R and S satisfy (10) for all $A \in T$.

Let $\mathcal{D}$ be the distribution of Gaussian random matrices of dimension $n \times r$ with i.i.d. entries $N(0,1)$ and let $G, H \sim \mathcal{D}$ be independent. It follows from Lemma 8 that with probability $\ge 1 - O(1)$,

$$\|\Sigma_R G H^T \Sigma_S^T\|_q \gtrsim \sqrt{r} E_q(R,S). \tag{12}$$

On the other hand, it follows from (10) that with probability $\ge 1 - \exp(-c_1 n)$,

$$\|\Sigma_R G H^T \Sigma_S^T\|_q \le D^{\frac{1}{2}-\frac{1}{p}} \|GH^T\|_p \lesssim D^{\frac{1}{2}-\frac{1}{p}} n r^{\frac{1}{p}}. \tag{13}$$

Now, let $\mathcal{F}$ be the distribution of an $n \times n$ Gaussian matrix of i.i.d. entries $N(0,1)$ and let F be drawn from $\mathcal{F}$. Then $\|RFS\|_q$ is identically distributed as $\Sigma_R F' \Sigma_S$, where F' is a random $t \times t$ Gaussian matrix of i.i.d. entries $N(0,1)$. It follows from Lemma 7 that with probability $\ge 1 - O(1)$,

$$\|\Sigma_R F' \Sigma_S^T\|_q \lesssim (\log^{\frac{5}{2}} t)(\log\log t) E_q(R,S) \le (\log^3 t) E_q(R,S) \tag{14}$$

On the other hand, it follows from (10) that with probability $\ge 1 - \exp(-c_2 n)$,

$$\|RFS^T\|_q \ge \|F\|_p \gtrsim n^{1/p}\sqrt{n}. \tag{15}$$

Define events $\mathrm{P}_1(G,H,R,S)$ and $\mathrm{P}_2(F,R,S)$ to be (12) and (14) respectively. Further define

$$\mathrm{P}_3(G,H):\ \left\|GH^T\right\|_p \lesssim n r^{1/p} \qquad \text{and} \qquad \mathrm{P}_4(F):\ \|F\|_p \lesssim n^{1/p}\sqrt{n}.$$

Both $\mathrm{P}_3(G,H)$ and $\mathrm{P}_4(F)$ hold with probability $\ge 1 - e^{-c_3 n}$ when $G, H \sim \mathcal{D}$ and $F \sim \mathcal{F}$.

Now, for $2m$ samples $G_1, \ldots, G_m, H_1, \ldots, H_m$ independently drawn from $\mathcal{D}$, and m samples $F_1, \ldots, F_m$ independently drawn from $\mathcal{F}$, it holds for any fixed R and S that

$$\Pr_{G_i,H_i,F_i}\{\exists i,\ \mathrm{P}_1(G_i,H_i,R,S), \mathrm{P}_2(F_i,R,S), \mathrm{P}_3(G_i,H_i), \mathrm{P}_4(F_i) \text{ all hold}\} \ge 1 - e^{-c_4 m}. \tag{16}$$

Since $1 \le \|R e_i e_j^T S^T\|_q = \|R_i\|_2 \|S_j\|_2 \le D$, we can restrict the matrix R and S to matrices with column norm in $[1, \sqrt{D}]$. Thus we can find a net $\mathcal{M} \subset \bigcup_{t=1}^{t_0} \mathbb{R}^{t \times n}$ of size $\exp(O(t_0 n \ln(Dn/\eta))$ such that for any M with column norms in $[1, \sqrt{D}]$, there exists $M' \in \mathcal{G}$ such that $\|M - M'\|_{op} \le \eta$.

Now it follows from (16) that

$$\Pr_{G_i,H_i,F_i}\{\forall R,S \in \mathcal{M}, \exists i,\ \mathrm{P}_1(G_i,H_i,R,S), \mathrm{P}_2(F_i,R,S), \mathrm{P}_3(G_i,H_i), \mathrm{P}_4(F_i) \text{ all hold}\}$$
$$\ge 1 - \exp\left(O\left(t_0 n \ln \frac{Dn}{\eta}\right)\right)\exp\left(-c_4 m\right) > 0,$$

provided that $m = \Theta(n \ln(Dn))$. Fix $\{G_i, H_i, F_i\}_i$ such that for each pair $R', S' \in \mathcal{M}$ there exists i such that $\mathrm{P}_1(G_i,H_i,R',S')$ and $\mathrm{P}_2(F_i,R',S')$ and $\mathrm{P}_3(G_i,H_i)$ and $\mathrm{P}_4(F_i)$ all hold.

Take $T = \{I_n\} \cup \{e_i e_j^T\}_{1 \le i,j \le n} \cup \{G_i H_i^T\}_{1 \le i \le m} \cup \{F_i\}_{1 \le i \le m}$. We know that if (R,S) satisfies (10) for all $A \in T$, then there exists R' and S' such that $\|R' - R\|_{op} \le \eta$ and $\|S' - S\|_{op} \le \eta$, and there exists $1 \le i \le m$ such that $\mathrm{P}_1(G_i,H_i,R',S')$ and $\mathrm{P}_2(F_i,R',S')$

and $\mathrm{P}_3(G_i, H_i)$ and $\mathrm{P}_4(F_i)$ all hold. One can then show that (12), (15) hold with slightly smaller constants and (13), (14) with slightly larger constants for R and S. It follows that

$$\frac{n^{\frac{1}{p}}\sqrt{n}}{\log^3 t} \lesssim D^{\frac{1}{p}-\frac{1}{2}}\sqrt{r}nr^{\frac{1}{p}}, \quad \text{or,} \quad \frac{1}{\log^3 t} \lesssim \left(\frac{rD}{n}\right)^{\frac{1}{p}-\frac{1}{2}} = \frac{1}{\rho^{\frac{2}{p}-1}},$$

which contradicts our choice of ρ (the hidden constant in $\lesssim$ above depends only on D_0, θ and η, and then we can choose the hidden constant in the Θ-notation for ρ). ◀

4 Upper Bounds

We shall only show the upper bounds for $1 \le p \le q \le 2$ in this section. Other cases can be found in the full version.

Let $G \in \mathbb{R}^{r\times n}$ ($r \ge Ct$) be a random matrix and $c, c', \eta > 0$ be absolute constants which satisfy the following properties:

(a) (subspace embedding) For a fixed t-dimensional subspace $X \subseteq \mathbb{R}^n$ it holds with probability $\ge 1 - \exp(-c't)$ that

$$(1-\eta)\|x\|_2 \le \|Gx\|_2 \le (1+\eta)\|x\|_2, \quad \forall x \in X;$$

(b) For a fixed $A \in \mathbb{R}^{n\times n}$ it holds with probability $\ge 1 - \exp(-c'r)$ that

$$\|GA\|_{op} \le c\left(\|A\|_{op} + \frac{1}{\sqrt{r}}\|A\|_F\right);$$

(c) For a fixed $A \in \mathbb{R}^{n\times n}$ it holds with probability $\ge 1 - \exp(-c'r)$ that

$$(1-\eta)\|A\|_F \le \|GA\|_F \le (1+\eta)\|A\|_F.$$

Consider the singular value decomposition $A = U\Sigma V^T$, where U and V are orthogonal matrices, $\Sigma = \mathrm{diag}\{\sigma_1, \ldots, \sigma_n\}$ with $\sigma_1 \ge \sigma_2 \ge \cdots$. For an index set $I \subseteq [n]$, define $A_I = U\Sigma_I V^T$, where Σ_I is Σ restricted to the diagonal elements with indices inside I (the diagonal entries with indices outside I are replaced with 0).

▶ **Theorem 9.** *Let $1 \le p \le q \le 2$. There exist constants $\theta = \theta(p,q) < 1$ small enough and $C = C(p,q)$ large enough such that for $t \le \theta n$ and matrix G satisfying the aforementioned properties, it holds for any (fixed) $A \in \mathbb{R}^{n\times n}$ with probabilty $1 - \exp(-c''t)$ that*

$$\frac{t^{\frac{1}{q}-\frac{1}{2}}}{n^{\frac{1}{p}-\frac{1}{2}}\log\frac{n}{t}}\|A\|_p \lesssim \|GA\|_q \lesssim \|A\|_p.$$

Note that for $t = \Omega(\log n)$ and $r = Ct$ for some large constant C, a Gaussian random matrix of i.i.d. entries $N(0, 1/r)$, or a randomized Hadamard Transform matrix of $r = \Theta(t\,\mathrm{polylog}(t))$ rows, satisfies the three conditions on G [7]. The following corollary of Theorem 9 is immediate.

▶ **Corollary 10.** *Suppose that $1 \le p \le q$ and $c\log n \le t \le \theta n$ for some absolute constants $\theta \in (0,1)$ and $c \ge 1$. There exists (random) $G \in \mathbb{R}^{r\times m}$ with $r \gtrsim t$ such that with probability $\ge 1 - \exp(-c''t)$,*

$$\|A\|_p \le \|GA\|_q \lesssim \frac{n^{\frac{1}{p}-\frac{1}{2}}}{t^{\frac{1}{q}-\frac{1}{2}}}\left(\log\frac{n}{t}\right)\|A\|_p.$$

In particular when $p = q$,

$$\|A\|_p \le \|GA\|_p \lesssim \left(\frac{n}{t}\right)^{\frac{1}{p}-\frac{1}{2}}\left(\log\frac{n}{t}\right).$$

Now we prove Theorem 9. We first need an auxiliary lemma.

▶ **Lemma 11.** *Let θ, t, C and G be as defined in Theorem 9 and $b = \Theta(\log(n/t))$. At least one of the following conditions will hold:*

$$\sum_{i=1}^{bt} \sigma_i^p \geq \frac{1}{2} \sum_{i=1}^{n} \sigma_i^p \tag{17}$$

and

$$\sigma_s^2 \leq \frac{2}{t} \sum_{i=s}^{n} \sigma_i^2 \quad \text{for some} \quad s \leq bt. \tag{18}$$

To prove the preceding lemma, consider the first b blocks of singular values of A each of size t, that is, $I_1 = \{\sigma_1, \ldots, \sigma_t\}$, $\ldots$, $I_b = \{\sigma_{(b-1)t+1}, \ldots, \sigma_{bt}\}$.

▶ **Lemma 12.** *If (18) does not hold for any $s \leq bt$, it must hold for all $2 \leq j \leq b$ that $\sigma_{jt} \leq \frac{1}{2}\sigma_{(j-1)t}$.*

Proof. If this is not true for some j then $\sum_{i=(j-1)t+1}^{jt} \sigma_i^2 \geq t\sigma_{jt}^2 > \frac{t}{2}\sigma_{(j-1)t}^2$, which contradicts (18) with $s = (j-1)t \leq bt$. ◀

Proof of Lemma 11. Suppose that (17) does not hold and we need to show that (18) holds for some $s \leq bt$. Otherwise, it follows from Lemma 12 that $\sigma_{bt+1} \leq \frac{\sigma_1}{2^b} \leq \left(\frac{t}{n}\right)^2 \sigma_1$ and thus

$$\sum_{i=bt+1}^{n} \sigma_i^p < n\sigma_{bt+1}^p \leq \frac{t^{2p}}{n^{2p-1}}\sigma_1^p \leq t\theta^{2p-1}\sigma_1^p, \tag{19}$$

On the other hand,

$$\sum_{i=1}^{bt} \sigma_i^p \geq t\sigma_1^p \left(\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^b}\right) = \left(1 - \frac{1}{2^b}\right) t\sigma_1^p = (1-\theta^2)t\sigma_1^p. \tag{20}$$

Using the assumption on θ, we see that the rightmost side of (20) is bigger than the rightmost side of (19), which contradicts the assumption that (17) does not hold. ◀

▶ **Lemma 13.** *Let $1 \leq p \leq q \leq 2$, and t, b and G be defined as in Lemma 11. Suppose that s satisfies (18) and let $J = \{s, s+1, \ldots, n\}$. Then*

$$\|GA_J\|_q \gtrsim \frac{t^{\frac{1}{q}-\frac{1}{2}}}{n^{\frac{1}{p}-\frac{1}{2}}} \|A_J\|_p.$$

Proof. Combining Property (b) of G with (18) yields that

$$\|GA_J\|_{op} \leq \frac{c}{\sqrt{t}} \left(\sqrt{2} + \sqrt{\frac{1}{C}}\right) \|A_J\|_F =: \frac{K}{\sqrt{t}} \|A_J\|_F$$

On the other hand, Property (c) states that $\|GA_J\|_F \geq \frac{1}{2}\|A_J\|_F$. This implies that at least αr singular values of GA_J are at least $\frac{\gamma}{\sqrt{t}}\|A_J\|_F$, provided that $C\left((1-\alpha)\gamma^2 + \alpha K^2\right) < \frac{1}{4}$, which is satisfied if we choose $\gamma \simeq 1/\sqrt{C}$ and $\alpha \simeq 1/K^{2/q}$. It follows that

$$\|GA_J\|_q \geq (\alpha r)^{\frac{1}{q}} \frac{\gamma}{\sqrt{t}} \|A_J\|_F \geq (\alpha C)^{\frac{1}{q}} \gamma \cdot \frac{t^{\frac{1}{q}-\frac{1}{2}}}{n^{\frac{1}{p}-\frac{1}{2}}} \|A_J\|_p.$$ ◀

▶ **Lemma 14.** *Let $1 \le p \le q \le 2$, and t, b and G be defined as in Lemma 11. Suppose that s satisfies* (18) *and let $J = \{s, s+1, \ldots, n\}$. Then*

$$\|GA_J\|_q \lesssim \frac{1}{t^{\frac{1}{p}-\frac{1}{q}}} \|A_J\|_p$$

Proof. When $p \le 2$, it holds that $\|A_J\|_F^2 \le \|A_J\|_p^p \|A_J\|_{op}^{2-p}$. Using (18), we obtain that

$$\|A_J\|_p \ge \frac{\|A_J\|_F^{2/p}}{\|A_J\|_{op}^{2/p-1}} \ge \left(\frac{t}{2}\right)^{\frac{1}{p}-\frac{1}{2}} \|A_J\|_F.$$

On the other hand, it follows from Property (c) of G that

$$\|GA_J\|_q \le r^{\frac{1}{q}-\frac{1}{2}} \|GA_J\|_F \le (1+\eta) r^{\frac{1}{q}-\frac{1}{2}} \|A_J\|_F.$$

Therefore,

$$\|GA_J\|_q \le (1+\eta) r^{\frac{1}{q}-\frac{1}{2}} \left(\frac{2}{t}\right)^{\frac{1}{p}-\frac{1}{2}} \|A_J\|_p = (1+\eta)(Cbqt)^{\frac{1}{q}-\frac{1}{2}} \left(\frac{2}{t}\right)^{\frac{1}{p}-\frac{1}{2}} \|A_J\|_p \lesssim \frac{1}{t^{\frac{1}{p}-\frac{1}{q}}} \|A_J\|_p. \blacktriangleleft$$

Now we are ready to show Theorem 9.

Proof of Theorem 9. It follows from the subspace embedding property of G that

$$(1-\eta)\|A_{I_i}\|_q \le \|GA_{I_i}\|_q \le (1+\eta)\|A_{I_i}\|_q, \quad 1 \le i \le b$$

and thus

$$\frac{1-\eta}{t^{\frac{1}{p}-\frac{1}{q}}} \|A_{I_i}\|_p \le \|GA_{I_i}\|_q \le (1+\eta)\|A_{I_i}\|_p.$$

When (17) holds, there exists i^* $(1 \le i^* \le b)$ such that

$$\|A_{I_{i^*}}\|_p \ge \frac{1}{2^{\frac{1}{p}} b} \|A\|_p$$

and thus

$$\frac{1}{bt^{\frac{1}{p}-\frac{1}{q}}} \|A\|_p \lesssim \|GA_{I_{i^*}}\|_q \lesssim \|A\|_p.$$

When (17) does not hold, let J be as defined in Lemma 13 and

$$\frac{1}{2^{\frac{1}{p}}} \|A\|_p \le \|A_J\|_p \le \|A\|_p.$$

The claimed upper and lower bounds follow from combining the bounds above, together with Lemma 13, Lemma 14, and

$$\max\left\{\|GA_{I_1}\|_q, \ldots, \|GA_{I_b}\|_q, \|GA_J\|_q\right\} \le \|GA\|_q \le \sum_{i=1}^{b} \left\|GA_{[I_i]}\right\|_q + \|GA_J\|_q,$$

noticing that $t^{\frac{1}{2}-\frac{1}{p}} / n^{\frac{1}{2}-\frac{1}{p}} \le 1/t^{\frac{1}{p}-\frac{1}{q}}$. ◀

References

1. Alexandr Andoni. Nearest neighbor search in high-dimensional spaces. In *the workshop: Barriers in Computational Complexity II*, 2010. `http://www.mit.edu/~andoni/nns-barriers.pdf`.
2. Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. Sketching and embedding are equivalent for norms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 479–488. ACM, 2015.
3. Jaroslaw Blasiok, Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. Streaming symmetric norms via measure concentration. arXiv:1511.01111 [cs.DS], 2016.
4. Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. Sketches for matrix norms: Faster, smaller and more general. arXiv:1609.05885 [cs.DS], 2016.
5. Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.
6. Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.
7. Michael B. Cohen, Jelani Nelson, and David P. Woodruff. Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 11:1–11:14, 2016.
8. Aram W. Harrow, Ashley Montanaro, and Anthony J. Short. Limitations on quantum dimensionality reduction. In *International Colloquium on Automata, Languages, and Programming*, pages 86–97. Springer, 2011.
9. Weihao Kong and Gregory Valiant. Spectrum estimation from samples. arXiv:1602.00061 [cs.LG], 2016.
10. Kasper Green Larsen and Jelani Nelson. The Johnson-Lindenstrauss Lemma Is Optimal for Linear Dimensionality Reduction. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:11, 2016.
11. Michel Ledoux and Michel Talagrand. *Probability in Banach spaces*. Springer-Verlag, Berlin, 1991.
12. Yi Li, Huy L. Nguyen, and David P. Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1562–1581, 2014.
13. Yi Li and David P. Woodruff. On approximating functions of the singular values in a stream. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 726–739, 2016.
14. Yi Li and David P. Woodruff. Tight bounds for sketching the operator norm, schatten norms, and subspace embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, pages 39:1–39:11, 2016.
15. Françoise Lust-Piquard. Inégalités de khintchine dans c_p $(1 < p < \infty)$. *Comptes Rendus de l'Académie des Sciences – Series I – Mathematics*, 303:289–292, 1986.
16. Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 91–100, 2013.

17 Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P. Woodruff. Spectral sums beyond fast matrix multiplication: Algorithms and hardness. arXiv:1704.04163 [cs.DS], 2017.

18 Jelani Nelson and Huy L. Nguyen. OSNAP: faster numerical linear algebra algorithms via sparser subspace embeddings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 117–126, 2013.

19 Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of $\mathrm{tr}(F(A))$ via stochastic lanczos quadrature, 2016. URL: `http://www-users.cs.umn.edu/~saad/PDF/ys-2016-04.pdf`.

20 Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Yonina C. Eldar and Gitta Kutyniok, editors, *Compressed Sensing: Theory and Practice*, pages 210–268. Cambridge University Press, 2012.

21 Andreas J. Winter. Quantum and classical message identification via quantum channels. *Quantum Information & Computation*, 5(7):605–606, 2005.

22 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

On Fast Decoding of High-Dimensional Signals from One-Bit Measurements*

Vasileios Nakos†

Harvard University, Cambridge, MA, USA
vasileiosnakos@g.harvard.edu

Abstract

In the problem of one-bit compressed sensing, the goal is to find a δ-close estimation of a k-sparse vector $x \in \mathbb{R}^n$ given the signs of the entries of $y = \Phi x$, where Φ is called the measurement matrix. For the one-bit compressed sensing problem, previous work [32, 19] achieved $\Theta(\delta^{-2}k\log(n/k))$ and $\tilde{\mathcal{O}}(\frac{1}{\delta}k\log(n/k))$ measurements, respectively, but the decoding time was $\Omega(nk\log(n/k))$. In this paper, using tools and techniques developed in the context of two-stage group testing and streaming algorithms, we contribute towards the direction of sub-linear decoding time. We give a variety of schemes for the different versions of one-bit compressed sensing, such as the for-each and for-all versions, and for support recovery; all these have at most a $\log k$ overhead in the number of measurements and $\text{poly}(k, \log n)$ decoding time, which is an exponential improvement over previous work, in terms of the dependence on n.

1998 ACM Subject Classification F.2.0 [Anaysis of Algorithms and Problem Complexity] General

Keywords and phrases one-bit compressed sensing, sparse recovery, heavy hitters, dyadic trick, combinatorial group testing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.61

1 Introduction

1.1 Standard Compressed Sensing

The compressed sensing framework describes how to reconstruct a vector (signal) $x \in \mathbb{R}^n$ given the linear measurements $y = \Phi x$ where $\Phi \in \mathbb{R}^{m\times n}$ for some $m \ll n$. This is an under-determined system with n variables and m equations. In many applications, however, such as images, we know that the vector x can be approximated by a k-sparse vector in some known basis. In this case, the matrix Φ contains a sufficient amount of information to roughly recover x if m is large enough; in particular, as shown in [4, 5], the signal can be approximately reconstructed from $\Theta(k\log(n/k))$ measurements when Φ is a Gaussian matrix. In order to do this, however, one has to solve the non-convex program

$$\min\|x\|_0 \text{ s.t. } y = \Phi x$$

However, [9, 4] show it is possible to avoid the non-convex formulation and, alternatively, we can use Basis Pursuit (BP), which changes the objective to $\min\|x\|_1$, and still recover a decent approximation of x. This can be solved using linear programming.

Compressed sensing, and sparse recovery, have appeared to be very useful tools in many areas such as analog-to-digital conversion [25], threshold group testing [1], discrete

* A full version of the paper is available at https://arxiv.org/abs/1603.08585.
† Supported in part by ONR grant N00014-15-1-2388.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 61; pp. 61:1–61:14


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

signal processing [12], streaming algorithms [29] and bioinformatics [26]. Depending on the application, one cares about optimizing different parameters of interest (measurements, decoding time, encoding time, failure probability).

Often we distinguish between the for-all model (or uniform recovery) and the for-each model (non-uniform recovery). In the for-all model, a single matrix is picked, which allows reconstruction of all k-sparse vectors. Whereas in the for-each model, the measurements are chosen at random such that, for some error probability p, they will contain sufficient information to reconstruct a single vector x with probability at least $1 - p$. We note that all aforementioned papers refer to the for-all model.

Moreover, it is desirable to achieve *sublinear* decoding time. The state of the art for the for-each model is [15]. The authors there achieve $k \cdot \text{poly}(\log n)$ decoding time with $\Theta(k \log(n/k))$ measurements. The failure probability was improved later in [17] using a much more complicated scheme. In the for-all model, Gilbert et.al. [18] give the first algorithm that runs in sublinear time with a number of measurements $\mathcal{O}(k\text{poly}(\log n))$. Porat and Strauss devised a scheme under a slightly weaker recovery guarantee with $\mathcal{O}(k \log(n/k))$ measurements accompanied with the first sublinear decoding procedure running in time $\mathcal{O}(k^{1-\alpha} n^{\alpha})$, for any constant α [33]. Subsequent work [16], the authors manage to bring the dependence of the approximation ϵ fact down to the right order of ϵ^{-1} and achieve runtime $\text{poly}(k, \log n)$, when $\epsilon \leq (\frac{\log k}{\log n})^{\gamma}$, for any constant γ.

1.2 One-Bit Compressed Sensing

Often in applications compressed sensing measurements must be quantized, since the requirement of infinite precision is not realistic: any measurement must be mapped to a small finite value in some universe [3]. Thus, one-bit compressed sensing emphasizes the compressed aspect of compressed sensing; many algorithms for standard compressed sensing rely on infinite precision in their real-valued inputs, relying on more precision than real sensors are capable of returning, an assumption which is unrealistic for real-world applications. Moreover, in hardware implementations, for example, where quantizers are implemented using comparators to zero [3], there is need of quantization to one-bit measurements. Comparators are indeed fast, but they are expensive, so minimizing their usage is really important. Moreover, dynamic range issues are a smaller problem in the case of one-bit quantizers.

It is clear that quantization increases the complexity of the decoding procedure and, additionally, is irreversible: given $y = \text{sign}(\Phi x)$ it is impossible to get the exact vector back. Previous results inquired the case in which the quantization maps each coordinate to $\{-1, +1\}$, which means that we learn only the sign of each coordinate. First, it is not obvious whether there is sufficient information to reconstruct a signal given its one-bit measurements. Of course, since we cannot know the length of the signal, nor the exact signal (even if its length were given), we can ask the following question: can we find its direction?

The problem was first studied in the work of Boufounos and Baraniuk [3], where the authors suggest recovering the signal x by solving the optimization problem

$$\min \|x\|_1 \text{ s.t.: } y \odot Ax \geq 0, \|x\|_2 = 1,$$

where $\odot$ stands for the element-wise product between two vectors. The goal is to find a vector y on the unit sphere such that $\|y - \frac{x}{\|x\|_2}\|_2^2 \leq \delta$. It is clear that this relaxation requires solving a non-convex program, an obstacle which Laska et al. [28] tried to remedy by giving an optimization algorithm that finds a stationary point of the aforementioned program; both papers, however, do not provide provable guarantees for the number of measurements needed. An alternative formulation was studied in [24], which showed that the number

of measurements could be brought down to $\mathcal{O}(\delta^{-1}k\log n)$, but the main obstacle of the non-convex formulation remained. In [31] Vershyin and Plan gave the first computationally tractable algorithm for the problem of one-bit compressed sensing by designing a compressed sensing scheme that approximately recovers a k-sparse vector from $\mathcal{O}(\delta^{-5}k\log^2(\frac{n}{k}))$ one-bit measurements via a linear programming relaxation. Their techniques were based on random hyperplane tessellations; the main geometric lemma they needed was that $\mathcal{O}(k\log(n/k))$ random hyperplanes partition the set of k-sparse vectors with unit norm into cells, each one having small diameter. In [32] the same authors improved the number of measurements to $\mathcal{O}(\delta^{-2}k\log(\frac{n}{k}))$ by analyzing a simple convex program. Their results can also be generalized to other sparsity structures, where the crucial quantity that determines the number of measurements is the gaussian mean-width of the set of all unit vectors having a specific sparsity pattern. Last but not least, they manage to handle gaussian noise and, most importantly, adversarial bit flips, though with a small worsening in the dependence on δ in their number of measurements. In [19] a two-stage algorithm with $\tilde{\mathcal{O}}(\frac{1}{\delta}k\log(n/k))$ measurements and $\mathcal{O}(nk\log(n/k)+\frac{1}{\delta^5}(k\log(n/k))^5)$ decoding time was proposed. Apart from recovering the vector, other algorithms that recover only the support of the signal have been proposed; see for example [19, 22].

In [20] it is conjectured that even if the support of the vector is known, the dependence of the number of measurements on δ must be at least $\frac{1}{\delta}$. In order to circumvent this, alternative quantization schemes were proposed, with the most common being Sigma-Delta quantization [21, 27]. In [2] Baraniuk, Foucart, Needell, Plan and Wootters manage to bring the dependence on δ down to $\log(\frac{1}{\delta})$ if the quantizer is allowed to be adaptive and the measurements take a special form of threshold signs.

1.3 Group Testing

In the group testing problem, we have a large population, which consists of "items", with a known number of defectives. The goal is to find the defectives using as few tests as possible, where a test is just a query whether a certain subset of items contains at least one defective. The group testing problem was first studied by Dorfman in [13]. There are two types of algorithms for this problem, namely adaptive and non-adaptive. In the first case, the outcome of previous tests can be used to determine future tests, whereas in non-adaptive algorithms all tests are performed at the same time. Group testing has many applications in DNA library screening and detection of patterns in data; more can be found in [6], [8].

Any solution for the group testing problem corresponds to a binary matrix, where the number of rows equals the number of tests and the number of columns equals the cardinality of the population. Given such a matrix M and a vector x indicating the positions of the defectives, we should be able to identify x from Mx, where the addition here corresponds to the OR operation of Boolean algebra. Since decoding time is important, the brute-force algorithm that iterates over each possible subset in order to recognise the defective set does not suffice. However, one can design matrices such that the naive decoding algorithm, which eliminates items belonging to negative tests and returns all the other items, correctly identifies all defective items [14]. In the literature these matrices are known as k-disjunct matrices.

In this paper, we are also interested in the so-called two-stage group testing problem, where two stages are allowed: the first stage recognises a superset of the defectives, and the second stage, which is performed after seeing the results of the first stage, recognizes the exact set of the defectives by querying separately for each one. We refer to (k,l) two-stage group testing as the case when there are k defectives and the superset is allowed to have up to $k+l$ elements. In fact, this is equivalent to the existence of a matrix M such that given

Mx one can find a set S with $k+l$ elements such that all defectives are included in S. The same naive algorithm, which eliminates all items that belong to a negative test and returns all other items, will be used here. A matrix is called (k,l) list-disjunct if this algorithm finds a superset of the support with at most $k+l$ elements. The term 'list-disjunct' appeared in [23], although it was also studied before in [11] under the name of super-imposed codes, and in [34] under the name of list-decoding super-imposed codes. In [30] Ngo, Porat and Rudra give efficient and strongly explicit constructions of matrices that allow two-stage group testing, which are also error-tolerant, in the sense that they can correct e_0 false positives and e_1 false negatives in sub-linear time and additional $\Theta(e_0 + k \cdot e_1)$ tests. They also prove matching lower bounds for several cases, including the case that $k = \Theta(l)$.

A group testing scheme is a tuple (M, R), where M is a matrix in $\{0,1\}^{m\times n}$ and R a procedure that takes as input Mx and outputs a vector y. Depending on the guaratee we want, we will either refer to it as two-stage group testing or (one-stage) group testing. The worst-case running time of the procedure R corresponds to the decoding time of the scheme.

1.4 Our Results

The main goal of our work is to make algorithms for one-bit compressed sensing not only "data-efficient" but also computationally efficient. Thus, we try to understand under which conditions and which number of measurements sublinear decoding time is possible. In the for-each version of noisy one-bit compressed sensing we give a scheme with almost-optimal measurements and sublinear decoding time. We also focus on decoding noiseless signals and presents several results towards this direction. We first give a scheme with sublinear decoding time with a small overhead in the number of measurements, by connecting the problem with Combinatorial Group Testing. Second, we try to understand whether it is possible to achieve a for-all guarantee for one-bit compressed sensing, while still keeping sublinear decoding time. We answer this question in the affirmative if we are allowed to use $\mathcal{O}(k^2 \log n)$ measurements. Our techniques also give a scheme for support recovery that outperforms the one in [19] by being exponentially faster than it; one additional aspect of our scheme is that the measurement matrix is explicit, which means that we can compute it in polynomial time.

For the case of general vectors, we define the δ-ℓ_2/ℓ_2 guarantee: For a unit vector $x \in \mathbb{R}^n$ we say that a scheme satisfies the δ-ℓ_2/ℓ_2 guarantee for one-bit compressed sensing if the output satisfies

$$\|\hat{x} - x\|_2^2 \leq c\|x_{\mathrm{tail}(k)}\|_2^2 + \delta,$$

while $x_{\mathrm{tail}(k)}$ is the vector that occurs after zeroing out the biggest k coordinates of x in magnitude and c is some absolute constant.

In the support recovery problem, one wants to construct a matrix Φ, such that for all k-sparse x, one is able to recover the support of the vector x, given measurements $y = \mathrm{sign}(\Phi x)$.

We present the results that we have in greater detail in Tables 1 and 2. We note that the decoding time of each scheme is $\mathrm{poly}(k, \log n)$.

- δ-ℓ_2/ℓ_2 for-each one-bit Compressed Sensing from $\mathcal{O}(k \log(n/k) \cdot (\log k + \log\log(n/k)) + \delta^{-2}k)$ measurements.
- For-each one-bit Compressed Sensing (noiseless signals) from $\mathcal{O}(k \log n + \log_k n \cdot \log\log_k n + \delta^{-2}k)$ measurements. This extends the result of [32], as it manages to also decrease the number of measurements for the for-each version of the problem for a wide range of parameters.

Table 1 Comparison of recovery schemes for one-bit compressed sensing.

Algorithm	Measurements	Decoding-Time	Model	Noise
[32]	$\delta^{-6}k\log(\frac{n}{k})$	$\text{poly}(n)$	For-all	Type 1
[32]	$\delta^{-2}k\log(\frac{n}{k})$	$\text{poly}(n)$	For-each	Type 2
[19]	$\tilde{\mathcal{O}}(\delta^{-1}k\log(\frac{n}{k}))$	$\mathcal{O}(nk\log n)+\text{poly}(k,\log n)$	For-all	No
This paper	$\delta^{-2}k+k\log(\frac{n}{k})(\log k+\log\log(\frac{n}{k}))$	$\text{poly}(k,\log n)$	For-each	Type 3
This paper	$k\log n+\delta^{-2}k+\log_k n\cdot\log\log_k n$	$\text{poly}(k,\log n)$	For-each	No
This paper	$k^2\log n\log\log_k n+\delta^{-6}k\log(\frac{n}{k})$	$\text{poly}(k,\log n)$	For-all	No

Table 2 Comparison of schemes for support recovery.

Algorithm	Measurements	Decoding time	Model
[19]	$k^3\log n$	$nk\log n$	For-all
This paper (Theorem 18)	$k^3\log n$	$k^3\text{poly}(\log n)$	For-all

- For-all one-bit Compressed Sensing (noiseless signals) in $\mathcal{O}(k^2\log n\log\log_k n+\delta^{-2}k\log n)$ measurements. This is the first scheme that allows sublinear decoding time in the for-all model of one-bit compressed sensing, although the dependence on k is k^2.
- Support recovery from one-bit measurement (noiseless signals) in $\mathcal{O}(k^3\log n)$ measurennts. This scheme is not only exponentially faster than the one presented in [19], but also explicit, in the sense that the matrix can be computed in polynomial time in n.

An interesting aspect of our results is that in the for-each model, the δ factor does not need to multiply the $k\log n$ factor, in contrast to the for-all version. We explain the three types of noise handled by the schemes in Table 1:

- Type 1 stands for adversarial bit flips. This means that after receiving $y=\text{sign}(\Phi x)$, an adversary can flip some of the entries of y, and then give it to the decoder. Here, we assume that x is exactly k-sparse. The result of [32] can tolerate up to $c\delta$ fraction of adversarial bit flips, where c is some absolute constant smaller than 1.
- Type 2 noise stands for gaussian random noise that is added to the k-sparse vector x after the matrix Φ has been applied to it. This means that $y=\text{sign}(Ax+u)$, where $u\sim c\mathcal{N}(0,I)$, where c is some absolute constant.
- Type 3 noise refers to general noise and is handled by the δ-ℓ_2/ℓ_2 guarantee. This means that $y=\text{sign}(\Phi(x_{\text{head}(k)}+x_{\text{tail}(k)}))$, where we can view the term $x_{\text{tail}(k)}$ as pre-measurement adversarial noise.

The ideas that are used in this paper to obtain sublinear decoding time are based on ideas that appeared in [30], as well as the dyadic trick, which has appeared in the streaming literature in the context of the Count-Min Sketch [10]. As far as we know, our work is the first that looks at sublinear decoding time in the one-bit compressed sensing framework and even achieves less measurements in some cases. Last but not least, we believe that a strong point of our schemes is their simplicity.

2 Preliminaries

We define the sign function as $\text{sign}(z)=+1$, for $z\geq 0$ and $\text{sign}(z)=-1$ for $z<0$. For a vector x, we define $\text{sign}(x)_i=\text{sign}(x_i)$, for all $i\in[n]$.

Any one-bit compressed sensing scheme is defined by a pair $(\mathcal{D}, \mathrm{Dec})$ where $\mathcal{D}$ is a distribution over $\mathbb{R}^{m\times n}$ and Dec is an algorithm that takes input $\mathrm{sign}(\Phi x)$ for some $x \in \mathbb{R}^n$ and gives back a vector $\hat{x}$. We will refer to Dec either as the "decoder" or "decoding procedure". We may also slightly abuse notation and refer to the pair (A, Φ), where A is a matrix coming from some distribution $\mathcal{D}$. We use m to denote the number of measurements, and *decoding time* refers to the running time of *Dec*. We also define $\Sigma_k = \{x : \|x\|_0 \leq k, \|x\|_2 \leq 1\}$ to be the set of all k-sparse vectors contained in the unit ℓ_2 ball, and $\Sigma_k^1 = \{x : \|x\|_2 = 1, \|x\|_0 \leq k\}$ the set of unit norm vectors with at most k non-zero coordinates.

For $x \in \mathbb{R}^n$ we denote its support set by $\mathrm{supp}(x)$. For a vector x, $\mathrm{head}(k)$ denotes the set of its k largest coordinates in magnitude, while $\mathrm{tail}(k)$ denotes the set of its $n-k$ smallest coordinates in magnitude. For a coordinate i, we say that i is a $\frac{1}{k}$-heavy hitter if, for some absolute constant C_h it holds that $x_i^2 \geq \frac{1}{C_h k}\|x_{\mathrm{tail}(k)}\|_2^2$. The constant C_h will be chosen later. For any x which we are sensing using a one-bit compressed sensing scheme, we assume that it $\|x\|_2 = 1$.

For each $S \subset n$, let $x_S \in \mathbb{R}^{|S|}$ denote the signal x restricted to coordinates in S. Similarly, for a matrix $M \in \mathbb{R}^{r\times n}$ and each $S \subset n$ let $M_S \in \mathbb{R}^{r\times|S|}$ be the matrix M restricted to columns in S.

▶ **Definition 1.** A scheme $(\mathcal{D}, \mathrm{Dec})$ satisfies the δ-ℓ_2/ℓ_2 guarantee for one-bit compressed sensing with failure probability p if for each $x \in \mathcal{S}^{n-1}$, it estimates a vector $\hat{x}$ such that

$$\forall x, P_{\Phi\sim\mathcal{D}}[\hat{x} = \mathrm{Dec}(\Phi x) : \|x - \hat{x}\|_2^2 \leq C\|x_{\mathrm{tail}(k)}\|_2^2 + \delta] \geq 1 - p,$$

where C is an absolute constant.

For function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $x \in \mathbb{R}^n$ we say that $y = f(x)$ is a sketch of x. In our case, f will always be of the form $f(x) = \mathrm{sign}(Ax)$, where $A \in \mathbb{R}^{m\times n}$.

We also give the definition of the tensor product of two matrices. We note that this is not the standard tensor product (or Kronecker product, as usually known) appearing in the literature.

▶ **Definition 2.** Let $A \in \{0,1\}^m \times \{0,1\}^N$ and $B \in \{0,1\}^{m'} \times \{0,1\}^N$. The tensor product $A \otimes B$ is an $mm' \times N$ binary matrix with rows indexed by the elements of $[m] \times [m']$ such that for $i \in [m]$ and $j \in [m']$, the row of $A \otimes B$ indexed by (i, j) is the coordinate-wise product of the i-th row of A and j- th row of B.

In order to proceed, we have to explain the difference between the for-all and the for-each model. Let P_x be the predicate that the sparse recovery scheme returns a vector $\hat{x}$ such that $\|x - \hat{x}\|_2^2 > \delta$, when the matrix Φ is chosen from the distribution $\mathcal{D}$. Let p be some target probability. In the for-each model the guarantee is that $\forall x \in \Sigma_k^1, \mathbb{P}[P_x] \leq p$. In the for-all model the guarantee is that $\mathbb{P}[\exists x \in \Sigma_k^1 : P_x] \leq p$. The randomness of the scheme is over the distribution $\mathcal{D}$.

The following result, appearing in [32] is a crucial component of most of our algorithms. This theorem is a special case of Theorem 1.1 from that paper, and it is also discussed in subsection 3.1 of the same paper (check "random noise before quantization" discussion).

▶ **Theorem 3.** *Let A be a random $m \times n$ matrix, with each entry being a standard gaussian, and all entries are independent. Let $x \in \Sigma_k^1$ and $y = \mathrm{sign}(Ax + v)$, where $v \sim \mathcal{N}(0, I)$. Then, the convex program*

$$z = \mathrm{argmax}\,\langle y, Az\rangle \;\; s.t. \;\; \|z\|_2 \leq 1, \|z\|_1 \leq \sqrt{k},$$

outputs a $\hat{x}$ such that $\|\hat{x} - x\|_2^2 \leq \delta$ with probability $1 - e^{-\Omega(k\log(n/k))}$, as long as $m = \Omega(\delta^{-2}k\log(n/k))$.

Algorithm 1 Naive Decoding Algorithm.

$S \leftarrow \emptyset$
for $i \in [n]$ **do**
  if exists no negative test where i participates in **then**
    $S \leftarrow S \cup \{i\}$
  end if
end for
Output S.

We should review some folklore definitions from Combinatorial Group Testing theory. For their proofs one can check [30].

▶ **Definition 4.** A $t \times n$ matrix M is k-disjunct if for every set $S \subset [n]$ with $|S| \leq k, \forall j \notin S, \exists i$ such that $M_{i,j} = 1$ but $\forall k \in S, M_{i,k} = 0$. In other words, $\text{supp}(M_j) - \cup_{l \in S}\text{supp}(M_l) \neq \emptyset$.

▶ **Definition 5.** A $t \times n$ matrix M is (k, l)-disjunct if for every two disjoint sets $S, T \subset [n]$ with $|S| \leq k, |T| = l$, there exists a row i such that $\forall j \in S, M_{i,j} = 0$, but $\exists j \in T, M_{i,j} = 1$.

In the noiseless case, we will make extensive use of the following two lemmas:

▶ **Lemma 6.** *Let M be a k-disjunct matrix. Then, given $y = Mx$, the naive decoding algorithm returns a set S such that $S = \text{supp}(x)$, i.e. the naive decoding algorithm correctly finds the support of x.*

▶ **Lemma 7.** *Let M be a (k, l)-disjunct matrix. Then, given $y = Mx$, the naive decoding algorithm returns a set S such that supp$(x) \subset S$ and $|S| \leq |\text{supp}(x)| + l$, i.e. the naive decoding algorithm finds a superset of the support of x with additional l elements.*

Of course, the two different definitions solve a different problem; the latter one solving a more relaxed version of Group Testing than the former. We will refer to the second version as two-stage group testing, whereas we will refer to the first version just as group testing.

2.1 Formal Statement of Results

▶ **Theorem 8.** *There exists a randomized construction of a scheme* $(\Phi,$ OneBitCS$())$ *which with probability* $1 - \mathcal{O}(\frac{1}{k \log(n/k)} + e^{-k})$ *satisfies the* δ-ℓ_2/ℓ_2 *guarantee. Moreover,* Φ *has* $\mathcal{O}(k \log(n/k)(\log k + \log\log(n/k)) + \delta^{-2}k)$ *rows and* OneBitCS$()$ *runs in time* $\text{poly}(k \log n)$.

▶ **Theorem 9.** *There exists a randomized construction of a scheme* $(\Phi,$ OneBitCS$())$ *such that*

$$\forall x \in \Sigma_k^1, \mathbb{P}[\hat{x} = \text{OneBitCS}(\Phi x), \|x - \hat{x}\|_2^2 > \delta] \leq e^{-k}.$$

The number of rows of Φ *is* $\mathcal{O}(k \log n + \log_k n \log\log_k n + \delta^{-2}k)$ *and the running time of* OneBitCS$()$ *is* $\text{poly}(k, \log n)$.

▶ **Theorem 10.** *There exists a randomized construction of a scheme* $(\Phi,$OneBitCS$())$ *such that*

$$\mathbb{P}[\exists x \in \Sigma_k^1 : \hat{x} = \text{OneBitCS}(\Phi x), \|x - \hat{x}\|_2^2 > \delta] \leq e^{-k \log(n/k)}.$$

The matrix Φ *has* $\mathcal{O}(k^2 \log(n/k) \log\log_k n + \delta^{-6}k \log(n/k))$ *rows and* OneBitCS$()$ *runs in poly*$(k, \log n)$ *time.*

▶ **Theorem 11.** *There exists a deterministic construction of a scheme* $(\Phi, \text{Support}())$*, where* $\text{Support}(\Phi x)$ *outputs a set* L *of at most* k *elements, and* $\forall x \in \Sigma_k^1, \text{Support}(\Phi x) = \text{supp}(x)$*. The matrix* Φ *has* $\mathcal{O}(k^3 \log n)$ *rows and* $\text{Support}()$ *runs in* $\text{poly}(k, \log n)$ *time.*

Complete proofs of these theorems are deferred to the full version.

2.2 Overview of techniques

All of our algorithms find the superset of the support of the vector x (noiseless case) or a small set containing the largest $\mathcal{O}(k)$ in magnitude coordinates (noisy case). This set contains the crucial information needed to approximate x. Then, by restricting to the set obtained, we show how the algorithm from [32] can give us the desired guarantees.

We first treat the noisy case. We sketch our approach to find a set of size $O(k)$ that contains all $\frac{1}{k}$-heavy hitters of our vector x. Let us first discuss the Count-Sketch [7] for finding heavy hitters in data streams (where we also have magnitude information). The Count-Sketch consists of $\log n$ different iterations: in each iteration r we hash every element to $\mathcal{O}(k)$ buckets using a 2-wise independent hash function $h_r : [n] \to [\mathcal{O}(k)]$, combined with random signs. This means that for each pair (r, b) the (r, b)-th measurement is $\sum_{i:h_r(i)=b} \sigma_{i,r} x_i$, where $\sigma_{i,r}$ are pairwise independent random signs. For each coordinate i and iteration r, we read the value of the bucket $h_r(i)$ multiplied by $\sigma_{i,r}$ to get an estimate for x_i. The median of all $\log n$ different estimates is our final estimate for x_i. This approach essentially solves a harder problem called ℓ_∞/ℓ_2 sparse recovery, but at this point we are interested in finding only the heavy hitters of x, not approximating their values; we will use the algorithm of [32] for that later. So, when we only have access to one-bit measurements an immediate approach is the following. Using the same hashing scheme as Count-Sketch, the decoding algorithm for every coordinate i and every iteration r checks if $\sigma_{i,r}$ agrees or disagrees with the sign of the value of the bucket $h_r(i)$, let this be $C_{r,h_{r(i)}}$. If this happens more than $\frac{3}{4}$ of the time we classify i as a heavy hitter. This happens because if i is a heavy hitter, with constant probability x_i will dominate the value of $h_r(i)$. Unfortunately, this does not suffice to give us sublinear decoding time.

What we need is a technique called the dyadic trick, which was introduced in [10], for the ℓ_1 case when all x_i are non-negative. In this case, we form a tree of size depth $\log n$, where level l corresponds to the decomposition of $[n]$ into 2^l equal-sized and disjoint intervals. The algorithm at each step keeps a list of size $\mathcal{O}(k)$ of "active" nodes, that is intervals that contain some heavy hitter. At every level we run a version of the Count-Min Sketch to find the heavy intervals (those that have ℓ_1 mass larger than $\frac{\|x_{\text{tail}(k)}\|_1}{k}$) and proceed by considering their children as active. The algorithm terminates when we reach the last level. Observe that at all times we want to guarantee that the list is of size $\mathcal{O}(k)$ and hence $\mathcal{O}(k \log n)$ nodes are visited in total.

In our case, however, we only have sign information about our measurements and we do not assume that x_i are non-negative. Remember that every node of the tree corresponds to an interval. For any interval I in each level of the tree we add a normal random variable in front of every x_i for $i \in I$ and when we hash nodes to buckets, we combine with a random sign. We explain what this means. Let us focus on some interval/node I. If we hash this node to U buckets using a hash function $h_r : [2^l] \to [U]$, then the contribution of the interval to the value of bucket $h_r(a)$ will be $\sigma_I \cdot \sum_{i \in I} g_i^I x_i$, where g_i^I is the gaussian corresponding to each $i \in I$ and σ_I is the sign associated with I. The idea is that $\sum_{i \in I} g_i^I x_i$ essentially approximates the ℓ_2 mass of the interval and hence one can expect $\sigma_I \cdot \sum_{i \in T} g_i^I x_i$ to be roughly $\sigma_I \cdot \|x_I\|_2$. This would mean that if I contains a heavy hitter, the sign of the

measurement it participates in should roughly be the same as the sign of $\sum_{i\in I} g_i^I x_i$ and hence, by repeating a lot of times, we can hope to clasify this interval as heavy. In this approach, however, there are technical hurdles, one of them being that we should not refresh the gaussians across iterations in the same level otherwise the signs will be uniformly at random. In the next section we show how to take care of all the details.

In the noiseless case, we use techniques and schemes developed in the context of two-stage group testing. More specifically, we show that if we take a (k, k)-disjunct matrix A and replace each non-zero entry with a normal random variable, we get a for-each scheme for identifying a superset S of the support of x. Using this idea and Lemma 7 we can detect a superset S of the support of x. If we also keep in parallel matrix G each entry of which is a normal random variable, we can use Theorem 3 by restricting G on the columns indexed by S to get the result of Theorem 9. We take a similar approach for the for-all version of the problem, namely Theorem 10. In this case we have to suffer an additional multiplicative factor of k in the measurements. More specifically, let $A \in \{0,1\}^{t\times n}$ be a (k, k)-disjunct matrix and let V be a $k \times n$ matrix. Then, the measurement matrix for is the vertical concatenation of $A \otimes V$ concatenated with G. For every i and x, $(a_i \otimes V)x$ can be seen as single test on x: $(a_i \otimes V)x = 0$ if and only if $\mathrm{supp(a_i)} \cap \mathrm{supp(x)} = \emptyset$. Vertically concatenating $(A \otimes V)$ and $((-A) \otimes V)$ we can check whether $(a_i \otimes V)x = 0$ or not. Then, a modification of Algorithm 1 and the for-all theorem of [32] (Result 1 from Table 1) gives us the desired result. For the support recovery problem, our approach is similar: We use a deterministic k-disjunct matrix A guaranteed by [23] and form the vertical concatenation of $A \otimes V$ and $(-A) \otimes V$. A similar reasoning as above, along with Lemma 6 gives the desired result. The reason we make use of a k-disjunct matrix and not a list-disjunct one is because we are interested in finding exactly the support.

3 For-each δ-ℓ_2/ℓ_2 One-Bit Compressed Sensing

As mentioned in the previous section, the algorithm is based on a two-stage approach. The first stage identifies the set S of the "heavy" coordinates of the vector x; these coordinates carry most of the ℓ_2 mass of x and hence the crucial information needed to approximate it. The second stage runs the convex program of [32] with the universe being S instead of $[n]$.

The most important part of the algorithm and essentially our contribution, that enables sublinear decoding time, is the procedure that finds the set S. As mentioned before, we turn our attention to the Count-Sketch and the dyadic trick [7, 10], and show that, with only a constant multiplicative increase in the measurement complexity, we can modify them so that they also work with one-bit measurements. We note that these algorithms appeared in the linear case of the very-relevant problem of finding heavy hitters in data streams.

In what follows, we assume that n is a power of 2. Let C_{-1}, C_0, C_1, C_2 be large enough constants to be defined later. For each l, we consider a partition of $[n]$ to 2^l equal-sized disjoint intervals and we denote by L_l^a the a-th interval in this partition. We also set $\Delta = \frac{1}{C_{-1}k\log n}, \Delta' = \log(\frac{1}{\Delta})$, where C_{-1} is an absolute constant larger than 1.

The sensing matrix Φ is the vertical concatenation of matrices $E^{(\log k)}, E^{(\log k+1)}, \ldots, E^{(\log n)}, A$. The number of rows of each $E^{(l)}$ is $C_0 \cdot C_1 \cdot C_2 \cdot k\Delta'$ and the number of rows of A is $\mathcal{O}(\delta^{-2}k)$. For $\log k \le l \le \log n$ let $E^{(l)}$ be the l-th matrix. $E^{(l)}$ consists of submatrices $E_1^{(l)}, \ldots, E_{C_2\Delta'}^{(l)}$. Each matrix $E_m^{(l)}$ consists of C_1 matrices $E_{m,t}^{(l)}$, $t = 1, \ldots, C_1$. Let $h_{l,m,t} : [2^l] \to [C_0k]$ be a hash function that maps intervals/nodes at level l to C_0k buckets. We define the q-th row of $E_{m,t}^{(l)}$ via its dot product with x:

$$\left\langle e_q^T E_{m,t}^{(l)}, x \right\rangle = \sum_{a:h_{l,m,t}(a)=q} \sigma_{m,t}^{l,a} \sum_{j\in L_l^a} g_{j,m}^{(l)} \cdot x_j,$$

OneBitHeavyHitters(y):

1. $S_{\log k-1} \leftarrow \{L^1_{\log k-1}, L^2_{\log k-1} \dots, L^{k/2}_{\log k-1}\}$
2. For $l = \log k$ to $\log n$:
3. For each L^i_{l-1} in S_{l-1} add L^{2i-1}_l and L^{2i}_l to S_l.
4. For every element L^a_l in S_l
5. If CheckIfHeavy(L^a_l) = 'light' remove L^a_l from S_l
6. Output every x in $S_{\log n}$

Figure 1 Recovery of ℓ_2 Heavy Hitters from One-Bit Measurements.

where $g^{(l)}_{j,m} \sim \mathcal{N}(0,1)$ and $\sigma^{l,a}_{m,t}$ are random signs. The above expression states that in each sketch $E^{(l)}_{m,t}$ we hash the nodes at level l in $C_2 k$ buckets.

In other words, every $E^{(l)}$ holds a hierarchical separation of $[n]$ into 2^l intervals of length $n/2^l$. Fix now some $m \in [C_2\Delta']$. Then, in each $E^{(l)}_{m,t}$, every node/interval is hashed to some bucket and the coordinates inside this interval are combined with standard Gaussians. Moreover, every interval is assigned a random sign. The intuition is that with constant probability, we do expect the term $\sum_{j \in L^a_l} g^{(l)}_{j,m,t} \cdot x_j$, to behave roughly like the l_2 mass of the interval itself. Then, by keeping the same gaussians, we take C_1 such hashing schemes (we refresh only the σ variables and the hash functions). For fixed m, l, this means that we use in total $C_1 C_2 \Delta'$ measurements, $C_2\Delta'$ for each of the C_1 rounds. Let this scheme be called Scheme 1. Then, for each level, we repeat the Scheme 1 $C_2\Delta'$ times, for $m = 1, \dots, C_2\Delta'$. Note now that across E^m_l for different m, l we use new g variables. The reason we have to make this additional repetition, in contrast to the standard dyadic trick, is that we only have sign information and we cannot use fresh gaussians at every measurement, since this would imply uniformity of the signs of the measurements. In other words, we would roughly see half $+1$ and half -1 and we would not be able to distinguish the 'heavy' intervals from the 'light' ones, as we will see next.

The decoding algorithm processes these intervals in increasing l for $l = \log k$ up to $\log n$ and keeps a list of intervals at each time (the list is denoted by S_l in the pseudocode). In the beginning of each step l, every node is hashed to $C_0 k$ buckets. Suppose for a moment, that we have the ℓ_2 mass of each interval and we hash these values, instead, into $C_0 k$ buckets combined with random signs. If an interval I contains a node that is 'heavy' and also is hashed to a bucket b, then we expect that its ℓ_2 mass dominates the ℓ_2 mass of other coordinates hashed to the same bucket. Thus, the sign of the sum must be determined by the sign of the 'heavy' interval. To overcome the fact that we do not have the ℓ_2 mass of the interval (since we can only make use of linear measurements) we add a standard random variable in front of every node in the inteval, before hashing. We exploit the aforementioned intuition, along with 2-stability of the Gaussian distribution , to show that we can identify all "heavy" intervals and that we do not introduce a big number of erroneous intervals (intervals that are not 'heavy'). We repeat this hashing scheme C_1 times with the same gaussians and try to find the intervals whose sign agrees or disagrees with the measurement they participate in most of the time. We consider them good. As mentioned in the previous paragraph, this whole hashing scheme called Scheme 1. Now, we repeat Scheme 1 $C_2\Delta'$ times with completely fresh randomness. We then find the intervals which were consider good at least $\frac{2}{3}C_2\Delta'$ times and add them to a list. At the end of each step l, every interval L^i_l that belongs to the list and is substituted by its two sub-intervals $L^{2i-1}_{l-1}, L^{2i}_{l-1}$.

CheckIfHeavy(L_l^a):

1. isheavy $\leftarrow 0$
2. For $m = 1$ to $C_2\Delta'$
3. cnt $\leftarrow 0$
4. For $t = 1$ to C_1
5. $y_q \leftarrow$ value of bucket $h_{l,m,t}(a)$
6. If $\text{sign}(y_q) = \sigma_{m,t}^{l,a}$
7. cnt $\leftarrow$ cnt $+ 1$
8. If cnt $> 0.8C_1$ *or* cnt $< 0.2C_1$
9. isheavy $\leftarrow$ isheavy $+ 1$
10. If isheavy $> \frac{2}{3}C_2\Delta'$
11. return 'heavy', else return 'light'

Figure 2 Check if an Interval at a Specific Level is Heavy.

OneBitCS(y):

1. $S \leftarrow$ OneBitHeavyHitters(y).
2. $\hat{x} = \text{argmax}\,\langle y, A_S z\rangle$ subject to $\|z\|_2 \leq 1, \|z\|_1 \leq \sqrt{k}$ (Algorithm of [32]).
3. Output $\hat{x}$.

Figure 3 One-Bit Compressed Sensing.

▶ **Definition 12.** For a coordinate i and a level l, let $b^l(i)$ be such that $i \in L_l^{b^l(i)}$. If the level l is implicit, we may simplify the notation to $b(i)$. In other words, $b^l(i)$ is the interval on level l which contains i.

Fix some matrix $E_{m_0}^l$ an an interval I in level l. Then, we will say that $E_{m_0}^l$ classifies interval I as good, if the variable isheavy is incremented in the execution of CheckIfHeavy(I) when $m = m_0$. Intuitively, $E_{m_0}^l$ classifies I as good if I appears to contain a heavy hitter in it. The next two lemmas are crucial components of our proof.

▶ **Lemma 13.** *Let C_h be an absolute constant. Fix m, l. Let $i \in [n]$ such that $|x_i|^2 > \frac{1}{C_h k}\|x_{\text{tail}(k)}\|_2^2$. Let I an interval at level l. Assume that $L_l^{b(i)}, I \in S_l$. Then, for some absolute constant c, the following claims hold:*

- *$E_m^{(l)}$ will classify $L_l^{b(i)}$ as good with constant probability, strictly larger than $\frac{1}{2}$.*
- *If there are at least ck intervals at the same level l which have greater ℓ_2 mass than I, then, with constant probabiliy, $E_m^{(l)}$ will not classify I as good.*

The proof of the aforementioned lemma is deferred to the full version.

▶ **Lemma 14.** *Let $S =$ OneBitHeavyHitters(y). then, with probability $1 - \mathcal{O}(\frac{1}{k\log(n/k)})$, for all $\log k \leq l \leq \log n$, the following holds for the set S_l:*

- *If $|x_i|^2 > \frac{1}{C_h k}\|x_{\text{tail}(k)}\|_2^2$ and $i \in L_l^{b(i)}$, then $i \in S$.*
- *$|S_l| \leq ck$, for some absolute constant c.*

Proof. For the proof of this lemma, we need to introduce some additional definitions. We will refer to any interval that contains a node i such that $|x_i|^2 > \frac{1}{10k}\|x_{\text{tail}(k)}\|_2^2$, as a type 1 interval. For a level l, we say that an interval at level l is of type 2, if there exist at least ck intervals at the same level that have greater ℓ_2 mass than this interval.

We now proceed by induction on the number of levels. The base case $l = \log k$ is trivial. We focus on some level l and assume that the induction hypothesis holds for all previous levels l. We prove the first bullet. Let i be a coordinate such that $|x_i|_2^2 > \frac{1}{C_h k}\|x_{\text{tail}(k)}\|_2^2$. By the induction hypothesis we get that $L_l^{b^l(i)} \in S_l$. From Lemma 14 we know that for any l, m, E_l^m will classify a type 1 interval as good with constant probability $> \frac{2}{3}$. Moreover, it will not classify any type 2 interval as good, again with constant probability $> \frac{2}{3}$. This implies that after repeating the same scheme $C_2\Delta' = C_2 \log(\frac{1}{\Delta})$ times, we will know, with probability at least $1 - \Delta$, if a specific interval is a type-1 interval or a type-2 interval or none of these. Because $\Delta = \Theta(\frac{1}{k \log(n/k)})$ we can take a union-bound over all possible intervals we might consider ($\mathcal{O}(k)$ at each of the $\log(n/k)$ levels), we can guarantee that every type-1 interval will remain in S_l, while any type-2 interval will be discarded from S_l. This implies that at any step we have at most ck intervals in S_l with every type-1 inteval belonging to S_l. ◀

We are now ready to prove the main result of this section.

Proof. By running OneBitCS(y), we obtain a set S that satisfies the guarantees of Lemma 14. Then, $y = \text{sign}(Ax) = \text{sign}(Ax_S + Ax_{[n]-S}) = \text{sign}(A_S x_S + v)$, where v is a vector each entry of which follows normal distribution with variance $\|x_{[n]-S}\|_2^2 \leq 1$. Clearly, A_S and v are independent and hence Theorem 3 applies. The number of rows needed equals $\Omega(\delta^{-2} k \log(ck/k)) = \Omega(\delta^{-2}k)$. The convex program of 3 outputs a vector $\hat{x}$ such that $\|\hat{x} - x_S\|_2^2 \leq \delta$. Since every coordinate i with $|x_i|^2 \geq \frac{1}{C_h k}\|x_{\text{tail}(k)}\|_2^2$ is contained in S,

$$\|x_{[n]-S}\|_2^2 \leq \|x_{\text{tail}(k)}\|_2^2 + ck\frac{1}{C_h k}\|x_{\text{tail}(k)}\|_2^2 = (1 + \frac{c}{C_h})\|x_{\text{tail}(k)}\|_2^2.$$

This implies that $\|x - \hat{x}\|_2^2 \leq (1 + \frac{c}{C_h})\|x_{\text{tail}(k)}\|_2^2 + \delta$, as desired. ◀

Acknowledgements. The author would like to thank Ely Porat for pointing him to [30], as well as Jelani Nelson for helpful discussions.

References

1 Rudolf Ahlswede, Lars Bäumer, Ning Cai, Harout K. Aydinian, Vladimir Blinovsky, Christian Deppe, and Haik Mashurian, editors. *General Theory of Information Transfer and Combinatorics*, volume 4123 of *Lecture Notes in Computer Science*. Springer, 2006.

2 Richard Baraniuk, Simon Foucart, Deanna Needell, Yaniv Plan, and Mary Wootters. Exponential decay of reconstruction error from binary measurements of sparse signals. *arXiv preprint arXiv:1407.8246*, 2014.

3 Petros Boufounos and Richard G. Baraniuk. 1-bit compressive sensing. In *42nd Annual Conference on Information Sciences and Systems, CISS 2008, Princeton, NJ, USA, 19-21 March 2008*, pages 16–21, 2008. doi:10.1109/CISS.2008.4558487.

4 E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, Dec 2006. doi:10.1109/TIT.2006.885507.

5 Emmanuel Candes, Mark Rudelson, Terence Tao, and Roman Vershynin. Error correction via linear programming. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 668–681. IEEE, 2005.

6 Fei-Huang Chang, Huilan Chang, and Frank K. Hwang. Pooling designs for clone library screening in the inhibitor complex model. *J. Comb. Optim.*, 22(2):145–152, 2011. doi:10.1007/s10878-009-9279-9.

7 Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

8 Hong-Bin Chen and Frank K. Hwang. A survey on nonadaptive group testing algorithms through the angle of decoding. *J. Comb. Optim.*, 15(1):49–59, 2008. doi:10.1007/s10878-007-9083-3.

9 Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

10 Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

11 Annalisa De Bonis, Leszek Gasieniec, and Ugo Vaccaro. Optimal two-stage algorithms for group testing problems. *SIAM Journal on Computing*, 34(5):1253–1270, 2005.

12 David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. doi:10.1109/TIT.2006.871582.

13 Robert Dorfman. The detection of defective members of large populations. *Ann. Math. Statist.*, 14(4):436–440, 12 1943. doi:10.1214/aoms/1177731363.

14 Ding-Zhu Du and Frank K. Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 1999.

15 Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. Approximate sparse recovery: optimizing time and measurements. *SIAM Journal on Computing*, 41(2):436–453, 2012.

16 Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. For-all sparse recovery in near-optimal time. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 538–550, 2014. doi:10.1007/978-3-662-43948-7_45.

17 Anna C. Gilbert, Hung Q. Ngo, Ely Porat, Atri Rudra, and Martin J. Strauss. l2/l2-foreach sparse recovery with low risk. In *Automata, Languages, and Programming*, pages 461–472. Springer, 2013.

18 Anna C. Gilbert, Martin J. Strauss, Joel A. Tropp, and Roman Vershynin. One sketch for all: fast algorithms for compressed sensing. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 237–246. ACM, 2007.

19 Sivakant Gopi, Praneeth Netrapalli, Prateek Jain, and Aditya Nori. One-bit compressed sensing: Provable support and vector recovery. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 154–162, 2013.

20 Vivek K. Goyal, Martin Vetterli, and Nguyen T. Thao. Quantized overcomplete expansions in IR N: analysis, synthesis, and algorithms. *Information Theory, IEEE Transactions on*, 44(1):16–31, 1998.

21 C Sinan Güntürk, Mark Lammers, Alex Powell, Rayan Saab, and Özgür Yilmaz. Sigma delta quantization for compressed sensing. In *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, pages 1–6. IEEE, 2010.

22 Ankit Gupta, Robert D. Nowak, and Benjamin Recht. Sample complexity for 1-bit compressed sensing and sparse classification. In *ISIT*, pages 1553–1557, 2010.

23 Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'10, pages 1126–1142, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1873601.1873692.

24 L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *Information Theory, IEEE Transactions on*, 59(4):2082–2102, April 2013. doi:10.1109/TIT.2012.2234823.

25 Laurent Jacques, Jason N. Laska, Petros T. Boufounos, and Richard G. Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transactions on Information Theory*, 59(4):2082–2102, 2013. doi:10.1109/TIT.2012.2234823.

26 Raghunandan M. Kainkaryam, Angela Bruex, Anna C. Gilbert, John Schiefelbein, and Peter J. Woolf. poolMC: Smart pooling of mRNA samples in microarray experiments. *BMC Bioinformatics*, 11:299, 2010. doi:10.1186/1471-2105-11-299.

27 Felix Krahmer, Rayan Saab, and Özgür Yilmaz. Sigma–delta quantization of sub-gaussian frame expansions and its application to compressed sensing. *Information and Inference*, page iat007, 2014.

28 J. N. Laska, Zaiwen Wen, Wotao Yin, and R. G. Baraniuk. Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements. *Signal Processing, IEEE Transactions on*, 59(11):5289–5301, Nov 2011. doi:10.1109/TSP.2011.2162324.

29 S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005. doi:10.1561/0400000002.

30 Hung Ngo, Ely Porat, and Atri Rudra. Efficiently decodable error-correcting list disjunct matrices and applications. *Automata, languages and programming*, pages 557–568, 2011.

31 Yaniv Plan and Roman Vershynin. One-bit compressed sensing by linear programming. *Communications on Pure and Applied Mathematics*, 66(8):1275–1297, 2013.

32 Yaniv Plan and Roman Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *Information Theory, IEEE Transactions on*, 59(1):482–494, 2013.

33 Ely Porat and Martin J. Strauss. Sublinear time, measurement-optimal, sparse recovery for all. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1215–1227. SIAM, 2012.

34 A. M. Rashad. Random coding bounds on the rate for list-decoding superimposed codes. *Problems of Control and Information Theory – Problemy Upravleniya i Teorii Informatsii*, 19(2):141–149, 1990.

String Inference from Longest-Common-Prefix Array*

Juha Kärkkäinen[1], Marcin Piątkowski[2], and Simon J. Puglisi[3]

1 **Helsinki Institute of Information Technology (HIIT), Helsinki, Finland; and Department of Computer Science, University of Helsinki, Helsinki, Finland**
juha.karkkainen@cs.helsinki.fi

2 **Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland**
marcin.piatkowski@mat.umk.pl

3 **Helsinki Institute of Information Technology (HIIT), Helsinki, Finland; and Department of Computer Science, University of Helsinki, Finland**
simon.puglisi@cs.helsinki.fi

Abstract

The suffix array, perhaps the most important data structure in modern string processing, is often augmented with the longest common prefix (LCP) array which stores the lengths of the LCPs for lexicographically adjacent suffixes of a string. Together the two arrays are roughly equivalent to the suffix tree with the LCP array representing the tree shape.

In order to better understand the combinatorics of LCP arrays, we consider the problem of inferring a string from an LCP array, i.e., determining whether a given array of integers is a valid LCP array, and if it is, reconstructing some string or all strings with that LCP array. There are recent studies of inferring a string from a suffix tree shape but using significantly more information (in the form of suffix links) than is available in the LCP array.

We provide two main results. (1) We describe two algorithms for inferring strings from an LCP array when we allow a generalized form of LCP array defined for a multiset of cyclic strings: a linear time algorithm for binary alphabet and a general algorithm with polynomial time complexity for a constant alphabet size. (2) We prove that determining whether a given integer array is a valid LCP array is NP-complete when we require more restricted forms of LCP array defined for a single cyclic or non-cyclic string or a multiset of non-cyclic strings. The result holds whether or not the alphabet is restricted to be binary. In combination, the two results show that the generalized form of LCP array for a multiset of cyclic strings is fundamentally different from the other more restricted forms.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Pattern Matching, G.2.1 [Combinatorics] Combinatorial Algorithms, G.2.2 [Graph Theory] Eulerian cycles

Keywords and phrases LCP array, string inference, BWT, suffix array, suffix tree, NP-hardness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.62

1 Introduction

For a string X of n symbols, the suffix array (SA) [23] contains pointers to the suffixes of X, sorted in lexicographical order. The suffix array is often augmented with a second array –

* The full version of this paper containing all the proofs and additional examples is available as [21], http://arxiv.org/abs/1606.04573.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 62; pp. 62:1–62:14

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

the longest common prefix (LCP) array – storing the length of the longest common prefix between lexicographically adjacent suffixes; i.e., LCP[i] is the length of the LCP of suffixes $X[\mathrm{SA}[i]..n)$ and $X[\mathrm{SA}[i-1]..n)$. The two arrays are closely connected to the suffix tree [32] – the compacted trie of all the string's suffixes: the entries of SA correspond to the leaves of the suffix tree, and the LCP array entries tell the string depths of the lowest common ancestors of adjacent leaves, defining the shape of the tree. For decades these data structures have been central to string processing; see [4] for a history and an overview, and [1, 3, 15, 30, 26] for further details on myriad applications.

Given both the suffix and the LCP array, the corresponding string is unique up to renaming of the characters and is easy to reconstruct: zeros in the LCP array tell where the first character changes in the lexicographical list of the suffixes, and the suffix array tells how to permute those first characters to obtain the string. Given the suffix array without the LCP array, we can easily reconstruct a corresponding string where all characters are different, and it is not difficult to characterize the set of all strings with a given suffix array [5, 28, 22]. In essence, the suffix array determines a set of positions in the LCP array that must be zero. Specifically, for any i let j and k be integers such that $\mathrm{SA}[j] = \mathrm{SA}[i-1]+1$ and $\mathrm{SA}[k] = \mathrm{SA}[i]+1$. Then, if $k < j$, we must have $\mathrm{LCP}[i] = 0$. For any other position, we can freely and independently decide whether the value is zero or not, and as described above, the zero positions together with the suffix array determine the string.

In this paper, we consider the problem of similarly reconstructing strings from an LCP array without the suffix array. As mentioned above, the LCP array determines the shape of the suffix tree, i.e., the suffix tree without edge or leaf labels. String inference from the suffix tree shape has recently been considered by three different sets of authors [19, 6, 31]. However, all of them assume that the suffix tree is augmented with significant additional information, namely *suffix links*, which makes the task much easier. Indeed, our new algorithms essentially reconstruct suffix links from the LCP array. According to Cazaux and Rivals [6], the case without suffix links was considered but not solved in [27]. We are also aware that others have considered it but without success [2].

To fully define the problem, we have to specify what kind of strings we are trying to infer. Often suffix trees and suffix arrays are defined for *terminated strings* that are assumed to end with a special symbol \$ that is different from and lexicographically smaller than any other symbol. The alternative is an *open-ended string* where no assumption is made on the last symbol. For suffix and LCP arrays the only change from omitting the terminator symbol is dropping the first element (which is always zero in the LCP array), but the suffix tree can change considerably because some suffixes can be prefixes of other suffixes and thus are not represented by a leaf. Inferring open-ended strings from a suffix tree (with suffix links) is studied by Starikovskaya and Vildhøj [31], who show that any string can be appended by additional characters without changing the suffix tree shape (thus the term open-ended). However, such an extension can change the suffix and LCP arrays a great deal, i.e., with the arrays a string is never truly open-ended but has at least an implicit terminator.

To get rid of even an implicit terminator, we consider a third type of strings, *cyclic strings*, where we use rotations in place of suffixes. For a terminated string, replacing suffixes with rotations causes no changes to the suffix/rotation array or the LCP array. Thus any integer array that is a valid LCP array for a terminated string is always a valid LCP array for a cyclic string too, but the opposite is not true. For example, the LCP array for the cyclic string *aababa* is $(2, 1, 3, 0, 2)$, which is not a valid LCP array for any non-cyclic string. In this sense, the cyclic string case is strictly more general. An even more striking example is a non-primitive string, such as *abab*, that has two or more identical rotations. For reasons

explained below, instead of rotations we use *cyclic suffixes* which are infinite repetitions of rotations. Thus the LCP array for the cyclic string $abab$ is $(\omega, 0, \omega)$, where ω denotes the positions of two adjacent identical cyclic suffixes.

As a further generalization, we may have a joint suffix array for a collection of strings, where we have all suffixes of all strings in lexicographical order, and the corresponding LCP array. In the terminated version, each string is terminated with a distinct terminator symbol. If we have an LCP array for a collection of open-ended strings, adding the terminator symbols simply prepends one zero for each terminator. The LCP array for a collection of terminated strings is identical to the LCP array of the concatenation of the strings. Thus the generalization from single strings to string sets does not add to the set of valid LCP arrays for terminated strings, but it does for cyclic strings. For example the LCP array for a string set $\{aa, b\}$ is $(\omega, 0)$, which is not a valid LCP array for any single string. For multiple cyclic strings, it is important to use cyclic suffixes instead of rotations because the result can be different (e.g., the set $\{ab, aba\}$).

Now we are ready to formally define the problem of String Inference from LCP Array (SILA). In the decision version, we are given an array of integers (and possibly ω's) and asked if the array is a valid LCP array of some string. If the answer is yes, the reporting version may also output some such string, and possibly a characterization of all such strings. Different variants are identified by a prefix: S for a string set; T, O, or C for terminated, open-ended or cyclic; and B for a binary alphabet (where terminators are not counted). For example, BCSSILA stands for Binary Cyclic String Set Inference from LCP Array. As discussed above, and summarized in the following result (see [21] for the proof), the non-cyclic variants are essentially equivalent, but the cyclic variants are more general.

▶ **Proposition 1.** *There are polynomial time reductions from BTSILA to BOSILA, BTSSILA, BOSSILA, TSILA, OSILA, TSSILA, and OSSILA.*

Our Contribution

Our first result is a linear time algorithm for BCSSILA. For a valid LCP array the algorithm outputs a string, which is the Burrows-Wheeler transform (BWT) of the solution string set. This relies on a generalization of the BWT for multisets of cyclic strings developed in [24, 20]. There can be more than one multiset of strings with the same BWT but the class of such string collections is simple and well characterized in [20]. The algorithm also outputs a set of substring swaps such that applying any combination of the swaps on the BWT produces another BWT of a solution, and any BWT of a solution can be produced by such a combination of swaps. Thus we have a complete characterization of all solutions. The number of swaps can be linear and thus the number of distinct solutions can be exponential. We also present an algorithm for CSSILA, i.e., without a restriction on the alphabet size, that has a polynomial time complexity for any constant alphabet size.

Our second result is a proof, by a reduction from 3-SAT, that (the decision version of) BCSILA, and thus CSILA, is NP complete. Therefore, even though the BCSSILA algorithm produces a characterization of all solutions, it is NP hard to determine whether one of the solutions is a single string. Furthermore, we modify the reduction to prove that BTSILA is NP complete too. By Proposition 1, this shows that all variants of SILA mentioned above except (B)CSSILA are NP complete. Since CSSILA is in P for constant alphabet sizes, this leaves the complexity of CSSILA for larger alphabets as an open problem.

Related Work

String inference from partial information is a classic problem in string processing, dating back some 40 years to the work of Simon [29], where reconstructing a string from a set of its subsequences is considered. Since then, string inference from a variety of data structures has received a considerable amount of attention, with authors considering border arrays [12, 11, 10], parameterized border arrays [18], the Lyndon factorization [25], suffix arrays [5, 22], KMP failure tables [11, 13], prefix tables [7], cover arrays [9], and directed acyclic word graphs [5]. The motivation for studying most string inference problems is to gain a deeper understanding of the combinatorics of the data structures involved, in order to design more efficient algorithms for their construction and use.

A (somewhat tangentially) related result to ours is due to He et al. [16], who prove that it is NP hard to infer a string from the longest-previous-factor (LPF) array. It is well known that LPF is a permutation of LCP [8] but otherwise it is a quite different data structure. For example, it is in no way concerned with lexicographical ordering. Like our NP-hardness proof, He et al.'s reduction is from 3-SAT, but the details of each reduction appear to be very different. Moreover, their construction requires an unbounded alphabet while our construction works for a binary alphabet and thus for any alphabet.

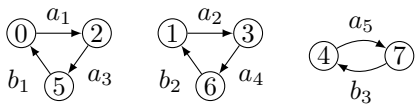
To the best of our knowledge, all of the previous string inference problems aim at obtaining a single non-cyclic string from some data structure, and we are the first to consider the generalizations to cyclic strings and to string sets, and as our results show, this makes a crucial difference. As explained in the next section, the generalizations arise naturally from the generalized BWT introduced in [24], which also played a central role in another recent result on the combinatorics of LCP arrays [20].

2 Basic notions

Let v be a string of length n and let $\widehat{v}$ be obtained from v by sorting its characters. The *standard permutation* [14, 17] of v is the mapping $\Psi_v : [0..n) \to [0..n)$ such that for every $i \in [0..n)$ it holds $\widehat{v}[i] = v[\Psi_v(i)]$ and for any $\widehat{v}[i] = \widehat{v}[j]$ the relation $i < j$ implies $\Psi_v(i) < \Psi_v(j)$. In other words, Ψ_v corresponds to the stable sorting of the characters. Let $C = \{c_i\}_{i=1}^{s}$ be the disjoint cycle decomposition of Ψ_v. We define the inverse Burrows–Wheeler transform IBWT as the mapping from v into a multiset of cyclic strings $W = \{\!\{w_i\}\!\}_{i=1}^{s}$ such that for any $i \in [1..s]$ and $j \in [0..|c_i|)$, $w_i[j] = v[\Psi_v(c_i[j])]$.

▶ **Example 2.** For $v = bbaabaaa$, we have $\mathrm{IBWT}(v) = \{\!\{aab, aab, ab\}\!\}$ as illustrated in the following table (showing $\widehat{v}$ and Ψ_v) and figure (showing the cycles of Ψ_v as a graph). The character subscripts are provided to make it easier to ensure stability.

i	0	1	2	3	4	5	6	7
$v[i]$	b_1	b_2	a_1	a_2	b_3	a_3	a_4	a_5
$\widehat{v}[i]$	a_1	a_2	a_3	a_4	a_5	b_1	b_2	b_3
$\Psi_v[i]$	2	3	5	6	7	0	1	4



The elements of W are primitive cyclic strings. *Cyclic* means that all rotations of a string are considered equal. For example, *aab*, *aba* and *baa* are all equal. A string is *primitive* if it is not a concatenation of multiple copies of the same string. For example, *aab* is primitive but *aabaab* is not. For any alphabet Σ, the mapping IBWT is a bijection between the set Σ^* of all (non-cyclic) strings and the multisets of primitive cyclic strings over Σ [24].

The set of positions of W is defined as the set of integer pairs $\mathrm{pos}(W) := \{\langle i,p\rangle \ : \ i \in [1..s],\ p \in [0..|w_i|)\}$. For a position $\langle i,p\rangle \in \mathrm{pos}(W)$ we define a *cyclic suffix* $W_{\langle i,p\rangle}$ as the infinite string that starts at $\langle i,p\rangle$, i.e., $W_{\langle i,p\rangle} = w_i[p]w_i[p+1 \bmod |w_i|]w_i[p+2 \bmod |w_i|],\ldots$. The multiset of all cyclic suffixes of W is defined as $\mathrm{suf}(W) := \{\!\{W_{\langle i,p\rangle} \ : \ \langle i,p\rangle \in \mathrm{pos}(W)\}\!\}$. We say that a string x occurs at position $\langle i,p\rangle$ in W if x is a prefix of the suffix $W_{\langle i,p\rangle}$.

The *(cyclic) suffix array* of a multiset of strings W is an array SA_W containing a permutation of $\mathrm{pos}(W)$ such that $W_{\mathrm{SA}_W[j-1]} \le W_{\mathrm{SA}_W[j]}$ for all $j \in [1..n)$. The *Burrows-Wheeler transform* (BWT) is a mapping from W into the string v defined as $v[j] = w_i[p-1 \bmod |w_i|]$, where $\langle i,p\rangle = \mathrm{SA}_W[j]$, i.e., $v[j]$ is the character preceding the beginning of the suffix $W_{\mathrm{SA}_W[j]}$. The BWT is the inverse of IBWT [24, 20].

The *longest-common-prefix array* $\mathrm{LCP}_W[1..n)$ is defined as $\mathrm{LCP}_W[j] = \mathrm{lcp}\big(W_{\mathrm{SA}_W[j-1]}, W_{\mathrm{SA}_W[j]}\big)$ for $0 < j < n$, where $\mathrm{lcp}(x,y)$ is the length of the longest common prefix between the strings x and y.

▶ **Example 3.** For $W = \{\!\{ab,\ aab,\ aab\}\!\}$ we have

$$\begin{aligned}\mathrm{suf}(W) &= \{\!\{(aab)^\omega, (aab)^\omega, (aba)^\omega, (aba)^\omega, (ab)^\omega, (baa)^\omega, (baa)^\omega, (ba)^\omega\}\!\}\\ \mathrm{SA}_W &= \big[\langle 2,0\rangle,\ \langle 3,0\rangle,\ \langle 2,1\rangle,\ \langle 3,1\rangle,\ \langle 1,0\rangle,\ \langle 2,2\rangle,\ \langle 3,2\rangle,\ \langle 1,1\rangle\big]\\ \mathrm{LCP}_W &= \big[\omega, 1, \omega, 3, 0, \omega, 2\big].\end{aligned}$$

The suffixes represented by the suffix array entries can also be expressed as follows.

▶ **Lemma 4.** *For $i \in [0..n)$, $W_{\mathrm{SA}_W[i]} = \widehat{v}[i]\widehat{v}[\Psi_v(i)]\widehat{v}[\Psi_v^2(i)]\widehat{v}[\Psi_v^3(i)]\ldots$.*

2.1 Intervals

Many algorithms on suffix arrays and LCP arrays are based on iterating over a specific types of array intervals. Next, we define these intervals and establish their key properties. For proofs and further details, we refer to [1, 26].

Let $v \in \{a,b\}^n$ and $W = IBWT(v)$. Let $\mathrm{SA} = \mathrm{SA}_W$ be the suffix array and $\mathrm{LCP} = \mathrm{LCP}_W$ the LCP array of W. Note that from now on, we will assume a binary alphabet.

▶ **Definition 5** (x-interval)**.** An interval $[i..j)$, $0 \le i \le j \le n$, is called the x-interval ($x \in \Sigma^*$) if and only if (1) x is not a prefix of $W_{\mathrm{SA}[i-1]}$ (or $i = 0$), (2) x is a prefix of $W_{\mathrm{SA}[k]}$ for all $k \in [i..j)$, and (3) x is not a prefix of $W_{\mathrm{SA}[j]}$ (or $j = n$).

In other words, in the suffix array the x-interval $\mathrm{SA}[i..j)$ consists of all suffixes of W with x as a prefix. Thus the size $j-i$ of the interval is the number of occurrences of x in W, which we will denote by n_x.

▶ **Definition 6** (ℓ-interval)**.** An interval $[i..j)$, $0 \le i < j \le n$, is called an ℓ-interval ($\ell \in \mathbb{N} \cup \{\omega\}$) if and only if (1) $LCP[i] < \ell$ (or $i = 0$), (2) $\min \mathrm{LCP}[i+1..j) = \ell$ (where $\min \mathrm{LCP}[j..j) = \omega$), and (3) $LCP[j] < \ell$ (or $j = n$).

▶ **Lemma 7.** *Every nonempty x-interval is an ℓ-interval for some (unique) $\ell \ge |x|$. Every ℓ-interval is an x-interval for some string x of length ℓ.*

▶ **Corollary 8.** *If an x-interval $[i..j)$ is an ℓ-interval for $\ell > |x|$, there exists a (unique) string y of length $\ell - |x|$ such that $[i..j)$ is the xy-interval.*

Thus the ℓ-intervals represent the set of all distinct x-intervals. This and the fact that the total number of ℓ-intervals is $\mathcal{O}(n)$ are the basis of many efficient algorithms for suffix arrays, see e.g., [1, 26].

Algorithm 1: Infer BWT from an LCP array.

Input: an array $\mathrm{LCP}[1..n)$ of integers and ω's
Output: a string $v \in \{a,b\}^n$ such that $\mathrm{LCP}_{\mathrm{IBWT}(v)} = \mathrm{LCP}$ together with a set S of swap intervals, or **false** if there is no such string v

```
1  S := ∅;
2  preprocess LCP for RMQs;
3  k := RMQ_LCP[1..n);
4  if LCP[k] ≠ 0 then
5      if LCP[k] = ω then return a^n, ∅;
6      else return false;
7  InferInterval([0,n), [0,k), [k,n));
8  compute W = IBWT(v), SA_W, and LCP_W;
9  if LCP_W ≠ LCP then return false;
10 return v, S;
```

3 Algorithm for BCSSILA

We are now ready to describe the algorithm for string inference from an LCP array. Given an LCP array $\mathrm{LCP}[1..n)$, our goal is to construct a string $v \in \{a,b\}^n$ such that $\mathrm{LCP} = \mathrm{LCP}_{\mathrm{IBWT}(v)}$. At first, we assume that such a string v exists, and consider later what happens if the input is not a valid LCP array.

Let $\mathrm{RMQ}_{\mathrm{LCP}}[i..j)$ denote the *range minimum query* over the LCP array that returns the position of the minimum element in $\mathrm{LCP}[i..j)$, i.e., $\mathrm{RMQ}_{\mathrm{LCP}}[i..j) = \arg\min_{k\in[i..j)} \mathrm{LCP}[k]$. The LCP array is preprocessed in linear time so that any RMQ can be answered in constant time (see for instance [26]). Then any x-interval can be split into two subintervals as shown in the following result.

▶ **Lemma 9.** *Let $[i..j)$ be an x-interval and an ℓ-interval for $\ell < \omega$, and let $k = \mathrm{RMQ}_{\mathrm{LCP}}[i+1..j)$. Then, for some string y of length $\ell - |x|$, $[i..k)$ is the xya-interval and $[k..j)$ is the xyb-interval.*

This approach makes it easy to recursively enumerate all ℓ-intervals. We will also keep track of ax- and bx-intervals together with any x-interval, even if we do not know x precisely. From the intervals we can determine the numbers of occurrences, n_{ax} and n_{bx}, which are useful in the inference of v:

▶ **Lemma 10.** *Let $[i..j)$ be the x-interval. Then $v[i..j)$ contains exactly n_{ax} a's and n_{bx} b's.*

In particular, when either n_{ax} or n_{bx} drops to zero, we have fully determined $v[i..j)$ for the x-interval $[i..j)$. In such a case, the LCP array intervals have to satisfy the following property.

▶ **Lemma 11.** *Let $[i_y..j_y)$ be the y-interval for $y \in \{x, ax, bx\}$. If $n_{ax} = j_{ax} - i_{ax} = 0$, then $\mathrm{LCP}[i_{bx}+1..j_{bx}) = 1 + \mathrm{LCP}[i_x+1..j_x)$, where $1+A$, for an array A, denotes adding one to all elements of A. Symmetrically, if $n_{bx} = 0$, then $\mathrm{LCP}[i_{ax}+1..j_{ax}) = 1 + \mathrm{LCP}[i_x+1..j_x)$.*

The main procedure is given in Algorithm 1. The main work is done in the recursive procedure InferInterval given in Algorithm 2. The procedure gets as input the x-, ax- and bx-intervals for some (unknown) string x, splits the x-interval into xya- and xyb-subintervals

Algorithm 2: InferInterval($[i_x..j_x)$, $[i_{ax}..j_{ax})$, $[i_{bx}..j_{bx})$).

Input: (nonempty) x-, ax- and bx-intervals
Output: Set $v[i_x..j_x)$ and add the swap intervals within $[i_x..j_x)$ to S

```
1  k_x := RMQ_LCP[i_x + 1..j_x); m_x := LCP[k_x];
2  if j_ax − i_ax = 1 then k_ax := i_ax; m_ax := ω;
3  else k_ax := RMQ_LCP[i_ax + 1..j_ax); m_ax := LCP[k_ax];
4  if j_bx − i_bx = 1 then k_bx := i_bx; m_bx := ω;
5  else k_bx := RMQ_LCP[i_bx + 1..j_bx); m_bx := LCP[k_bx];
6  if m_ax > m_x + 1 and m_bx > m_x + 1 then
7      if LCP[i_ax + 1..j_ax) = 1 + LCP[i_x + 1..k_x) then
8          v[i_x..k_x) = aa...a; v[k_x..j_x) = bb...b;
9          if LCP[i_ax + 1..j_ax) = 1 + LCP[k_x + 1..j_x) then add [i_x..j_x) to S ;
10     else
11         v[i_x..k_x) = bb...b; v[k_x..j_x) = aa...a;
12 else if m_ax > m_x + 1 then
13     if k_bx − i_bx = k_x − i_x then
14         v[i_x..k_x) = bb...b;
15         InferInterval([k_x..j_x), [i_ax..j_ax), [k_bx..j_bx));
16     else
17         v[k_x..j_x) = bb...b;
18         InferInterval([i_x..k_x), [i_ax..j_ax), [i_bx..k_bx));
19 else if m_bx > m_x + 1 then
20     if k_ax − i_ax = k_x − i_x then
21         v[i_x..k_x) = aa...a;
22         InferInterval([k_x..j_x), [k_ax..j_ax), [i_bx..j_bx));
23     else
24         v[k_x..j_x) = aa...a;
25         InferInterval([i_x..k_x), [i_ax..k_ax), [i_bx..j_bx));
26 else
27     InferInterval([i_x..k_x),[i_ax..k_ax),[i_bx..k_bx));
28     InferInterval([k_x..j_x),[k_ax..j_ax),[k_bx..j_bx));
```

based on Lemma 9, and tries to split ax- and bx-intervals similarly. If all subintervals are nonempty, the algorithm processes the two subinterval triples recursively (lines 27 and 28).

When trying to split the ax-interval, the result may be, for example, that the $axya$-interval is empty. In this case, we do not need to recurse on the xya-interval since the corresponding part of v must be all b's. The algorithm recognizes the emptiness of $axya$- or $axyb$-interval by the fact that $m_{ax} > m_x + 1$, but the problem is to decide which is the empty one. In most cases, this can be determined by comparing the sizes of the different subintervals or even the actual LCP-intervals (see Lemma 11).

There is one case, where the algorithm is unable to determine the empty subintervals, which is when $\mathrm{LCP}[i_{ax}{+}1..j_{ax}) = \mathrm{LCP}[i_{bx}{+}1..j_{bx}) = 1{+}\mathrm{LCP}[i_x{+}1..k_x) = 1{+}\mathrm{LCP}[k_x{+}1..j_x)$. Then, either the $axya$- and $bxyb$-intervals are empty or the $axyb$- and $bxya$-intervals are empty, but there is no way of deciding between the two cases. It turns out that both are valid

choices. The algorithm sets v according to one choice (line 8) but records the alternative choice by adding the interval to the set S. In such a case, the string xy is called a *swap core* and the xy-interval (equal to the x-interval) is called a *swap interval*.

For each swap interval $[i..j)$, the algorithm sets $v[i..k) = aa\dots a$ and $v[k..j) = bb\dots b$, where $k = (i+j)/2$, but swapping the two halves would be an equally good choice. Therefore, if the output of the algorithm contains s swap intervals, it represents a set of 2^s distinct strings. The following lemma shows that the swaps indeed do not affect the LCP array (the proof can be found in [21]).

▶ **Lemma 12.** *Let $v \in \{a,b\}^n$, $W = \mathrm{IBWT}(v)$, $\mathrm{SA} = \mathrm{SA}_W$ and $\mathrm{LCP} = \mathrm{LCP}_W$. Let x be a string that occurs in W and satisfies: (1) $\mathrm{LCP}[i_{xa}+1..j_{xa}) = \mathrm{LCP}[i_{xb}+1..j_{xb})$, and (2) $v[i_{xa}..j_{xa}) = aa\dots a$ and $v[i_{xb}..j_{xb}) = bb\dots b$, where $[i_z..j_z)$ is the z-interval for $z \in \{xa, xb\}$. Let v' be the same as v except that $v'[i_{xa}..j_{xa}) = bb\dots b$ and $v'[i_{xb}..j_{xb}) = aa\dots a$. Then $\mathrm{LCP}_{\mathrm{IBWT}(v')} = \mathrm{LCP}$.*

▶ **Theorem 13.** *Algorithm 1 computes in linear time a representation of the set of all strings $v \in \{a,b\}^*$ such that $\mathrm{LCP}_{\mathrm{IBWT}(v)}$ is the input array, or returns false if no such string exists.*

Proof. Since the algorithm verifies its result (lines 9 and 10), it will return false if the input is not a valid LCP array. Given a valid LCP array, Algorithm 2 sets all elements of v since it recurses on any subinterval that it doesn't set. All the choices made by the algorithm are forced by the lemmas in this and the previous section. The swap intervals record all alternatives in the cases where the content of v could not be fully determined, and all of those alternatives have the same LCP array by Lemma 12. It is also easy to see that the algorithm runs in linear time. ◀

4 Coupling Constrained Eulerian Cycle

We will now set out to prove the NP-completeness of the single string inference problems BCSILA and BTSILA. The proofs are done by a reduction from 3-SAT via an intermediate problem called Coupling Constrained Eulerian Cycle (CCEC) described in this section.

Consider a directed graph G of degree two, i.e., every vertex in G has exactly two incoming and two outgoing edges. If G is connected, it is Eulerian. An Eulerian cycle can pass through each vertex in two possible ways, which we call the *straight state* and the *crossing state* of the vertex as illustrated here:

We consider each vertex to be a *switch* that can be flipped between these two states. The combination of vertex states is called the *graph state*. For a given graph state, the paths in the graph form, in general, a collection of cycles. The Eulerian cycle problem can then be stated as finding a graph state such that there is only a single cycle; we call such a graph state Eulerian.

In the *Coupling Constrained Eulerian Cycle (CCEC) problem*, we are given a graph as described above, an initial graph state, and a partitioning of the set of vertices. If we flip a vertex state, we must simultaneously flip the states of all the vertices in the same partition, i.e., the vertices in a partition are coupled. A graph state that is achievable from the initial state by a set of such *partition flips* is called a *feasible state*. The CCEC problem is to determine if there exists a feasible graph state that is Eulerian.

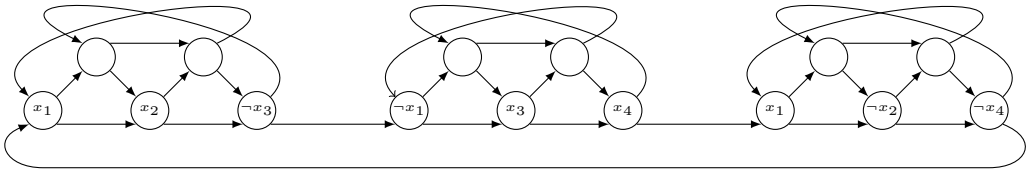


Figure 1 The CCEC graph corresponding to a 3-CNF formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_4)$.

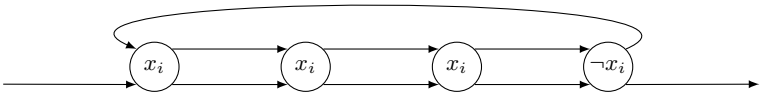
▶ **Theorem 14.** *CCEC is NP-complete.*

Proof. The proof is by reduction from 3-SAT. To obtain a CCEC graph from a 3-CNF formula, a gadget of five vertices is constructed from each clause and these gadgets are connected by a cycle. In each gadget, three of the vertices are labeled by the literals of the corresponding clause; the other two are called free vertices. See Fig. 1 for an illustration.

Each labeled vertex is in a straight state if the labeling literal is false and in a crossing state if the literal is true; their initial state corresponds to some arbitrary truth assignment to the variables. For each variable x_i, there is a vertex partition consisting of all vertices labeled by x_i or $\neg x_i$, so that flipping this partition corresponds to changing the truth value of x_i. Each free vertex forms a singleton partition and has an arbitrary initial state. Thus a graph state is feasible iff the labeled vertex states correspond to some truth assignment.

If a clause is false for a given truth assignment, the labeled vertices in the corresponding gadget are all in a straight state. This separates a part of the gadget from the main cycle and thus the graph state is not Eulerian. If a clause is true, at least one of the labeled vertices in the gadget is in a crossing state. Then we can always choose the state of the free vertices so that the full gadget is connected to the main cycle. Thus there exists a feasible Eulerian graph state iff there exists a truth assignment to the variables that satisfies all clauses. ◀

For purposes that will become clear later, we modify the above construction by adding some extra components to the graph without changing the validity of the reduction. Specifically, for each variable x_i in the 3-CNF formula we add the following gadget to the main cycle:



The vertices in the gadget are treated similarly to the other vertices in the graph: they belong to the partition with the other vertices labeled by x_i or $\neg x_i$, and the initial state is determined by the truth value of the labeling literal. It is easy to see that the gadget will be fully connected to the main cycle whether x_i is true or false. Thus the extra gadgets have no effect on the existence of an Eulerian cycle. Finally, we insert to the main cycle a single vertex labelled y with a self loop and forming a singleton partition.

5 BCSILA to CCEC

The next step is to establish a connection between the BCSILA and CCEC problems by showing a reduction from BCSILA to CCEC. Although the direction of the reduction is opposite to what we want, this construction plays a key role in the analysis of the main construction described in the next section.

Given a BCSILA instance (an integer array), we use Algorithm 1 to produce a representation of a set V of strings. The problem is then to decide if there exists $v \in V$ such that

IBWT(v) is a single (cyclic) string. We will write V as a string with brackets marking the swaps. For example, $V = b[ab][ab]a = \{bababa, babbaa, bbaaba, bbabaa\}$. In Example 2, we saw that the inverse BWT of a string $v \in V$ can be represented as a graph G_v where the vertices are labeled by positions in v and there is an edge between vertices i and j if, for some character $c \in \{a, b\}$ and some integer k, $\widehat{v}[i] = c$ is the kth occurrence of c in $\widehat{v}$ and $v[j] = c$ is the kth occurrence of c in v. Such an edge (i, j) is labeled by c_k. Note that $\forall v \in V$, $\widehat{v}$ is the same; we will denote it by $\widehat{V}$. We form a generalized graph G_V as a union of the graphs G_v, $v \in V$.

Consider a_k (the kth a) in $\widehat{V}$, say at position i. If a_k is outside any swap region in V, say at position j, there is a single edge (i, j) in G_V labeled by a_k. If a_k is within a swap region in V, it has two possible positions in the strings $v \in V$, say j and j'. That same pair of positions are also the possible positions of some b, say $b_{k'} = \widehat{V}[i']$. Then g_v has two edges, (i, j) and (i, j'), labeled with a_k and two edges, (i', j) and (i', j'), labeled with $b_{k'}$. The positions/vertices j and j' are called a *swap pair*.

To obtain a CCEC graph $\widetilde{G}_V$, we make two modifications to G_V. First, we merge each swap pair into a single vertex. Each merged vertex now has two incoming and two outgoing edges and all other vertices have one incoming and one outgoing edge. Second, we remove all vertices with degree one by concatenating their incoming and outgoing edges.

The initial state of the vertices in $\widetilde{G}_V$ is set so that the cycles in $\widetilde{G}_V$ correspond to the cycles in G_v for some $v \in V$. Two vertices in $\widetilde{G}_V$ belong to the same partition if their labels belong to the same swap interval in V. Then we have a one-to-one correspondence between swaps in V and partition flips in $\widetilde{G}_V$. If this CCEC instance has a solution, the Eulerian cycle spells a single string realizing the input LCP array. If the CCEC instance has no solution, the original BCSILA problem has no solution either.

6 BCSILA is NP-Complete

We are now ready to show that BCSILA is NP-complete using the reduction chain 3-SAT $\rightarrow$ CCEC $\rightarrow$ BCSILA. The first step was described in Section 4, and we will next describe the second. The latter reduction is not a general reduction from an arbitrary CCEC instance but works only for a CCEC instance obtained by the first reduction (including the extra gadgets).

The above BCSILA to CCEC reduction transforms each pair of swapped positions into a vertex and each swap interval into a vertex partition. Our construction creates a BCSILA instance such that the resulting BWT has the necessary swaps to produce the CCEC instance vertices and partitions. However, the BWT also has some unwanted swaps producing spurious vertices, but we will show that these spurious vertices do not invalidate the reduction.

Starting from a CCEC instance, we construct a set of cyclic strings and obtain the BCSILA instance as the LCP array of that string set. The construction associates two strings to each vertex and the cyclic strings are formed by concatenating the vertex strings according to the cycles in the graph in its initial state. The two passes of the cycles through a vertex must use different strings but it does not matter which pass uses which string.

Let n be the number of vertices in the CCEC graph and let m be the number of vertex partitions. We number the vertices from 1 to n and the partitions from 1 to m. The biggest partition number is assigned to the partition with the vertex y, the second biggest to the partition corresponding to the variable x_1, the third biggest to variable x_2, and so on. The three biggest vertex numbers are assigned to the vertices labeled x_1 in the extra gadget for the variable x_1, the next three biggest to the extra gadget vertices labeled x_2 and so on.

Within each extra gadget, the biggest number is assigned to the middle one of the three vertices. The strings associated with a vertex are ba^kba^{m+2h} and bba^kbba^{m+2h-1}, where k is the partition number and h is the vertex number. This completes the description of the transformation from a CECC instance to a BCSILA instance.

Let us now analyze the transformation by changing the BCSILA instance back to a CCEC instance using the construction of the preceding section. Specifically, we will analyze the swaps in the BWT produced from the LCP array. Let W be the set of cyclic strings constructed from the CCEC instance, and let V be the BWT with swaps constructed from LCP_W. An interval $[i..j)$ in V is a swap interval if and only if (1) $[i..j)$ is an x-interval for a string x such that either $occ(axa) = occ(bxb) = occ(x)/2$ or $occ(axb) = occ(bxa) = occ(x)/2$, where $occ(y)$ is the number of occurrences of y in W, and (2) $\mathrm{LCP}_W[i+1..k) = \mathrm{LCP}_W[k+1..j)$, where $k = (i+j)/2$. If $[i..j)$ is a swap interval, the string x is called its *swap core*. Our goal is to identify all swap cores.

Let us first consider strings of the form $x = ba^kb$. If $k > m$, $occ(x) \leq 1$ and x cannot be a swap core. For $k \in [1..m]$, x is always a swap core and corresponds to the CCEC partition numbered k. Let $v = \mathrm{BWT}(W)$ and let V' be v together with the swaps for cores of the form $x = ba^kb$, $k \in [1..m]$. It is easy to verify that a CCEC instance constructed from V' as described in the previous section is identical to the original CCEC instance. Thus, if there were no other swap cores, we would have a perfect reduction.

Unfortunately, there are other swap cores. A systematic examination of all strings (see [21] for details) shows that the other swap cores must be of the following forms: ba^{m+2n-1}, $a^{m+2n-1}b$, a^mba^m, a^mbba^m,a^kba^h, a^kbba^h, $a^kba^iba^h$ and $a^kbba^ibba^h$. Furthermore, it shows that each such swap core has exactly two occurrences, which means that the values k and/or h have to be sufficiently large. Each extra swap core adds a free vertex that is connected to the graph by making two existing edges to pass through the new vertex. Because of the way we chose to assign the biggest partition and vertex numbers, all the additional connections are within the extra gadgets, which does not change the existence of an Eulerian cycle. This completes the proof.

▶ **Theorem 15.** *BCSILA is NP-complete.*

7 BTSILA is NP-Complete

We will now show that BTSILA is NP-complete by modifying the above reduction for BCSILA to include a single terminator symbol \$ in the strings. The modification is applied to the set W of cyclic strings derived from the CCEC instance such that LCP_W is the BCSILA instance. Specifically, we replace the (unique) occurrence of a^{m+2n}, which is the longest consecutive run of a's, with $a^{m+2n+1}\$a^{m+2n}$ to obtain $W_\$$ and $\mathrm{LCP}_{W_\$}$. We will show that $\mathrm{LCP}_{W_\$}$ is a yes-instance of CSILA iff LCP_W is a yes-instance of BCSILA. Furthermore, if a cyclic string u is a solution to the CSILA instance, i.e., $\mathrm{LCP}_u = \mathrm{LCP}_{W_\$}$, then $\mathrm{LCP}_v = \mathrm{LCP}_{W_\$}$, where v is the rotation of u ending with \$ interpreted as a terminated string. Thus $\mathrm{LCP}_{W_\$}$ is a yes-instance of BTSILA iff it is a yes-instance of CSILA iff LCP_W is a yes-instance of BCSILA.

In general, adding even a single occurrence of a third symbol complicates the inference of the BWT from the LCP array and means that the set of equivalent BWTs can no more be described by a set of swaps. Consider how the operation of the procedure InferInterval (Algorithm 2) changes. First, it gets an extra $\$x$-interval as an input in addition to x-, ax- and bx-intervals. Second, the x-interval may be split into three subintervals, $xy\$$-, xya- and xyb-intervals, instead of two (which happens when the LCP interval contains two identical

minima). This leads to many more combinations to consider, and some of those combinations are more complicated.

Fortunately, in our case, having the single \$ surrounded by the two longest runs of a's simplifies things, and we will describe a modification of InferInterval to handle this case. Every call to InferInterval belongs to one of the following three types: (1) the x-interval is split into two and the $\$x$-interval is empty, (2) the x-interval is split into two and the $\$x$-interval is non-empty, and (3) the x-interval is split into three. The first case needs no modification at all. The other two cases mean that either $\$x$ or $x\$$ occurs in the produced string set, and since this property is not affected by swaps (or the threeway permutations described below), one of them occurs in every produced string set including $W_\$$. Since x must occur at least twice, one of the latter two cases happens iff $x = a^k$ for some $k \in [0..m + 2n]$. Although in general InferInterval cannot always know x, it is easy to keep track of x when $x = a^k$.

When InferInterval is called with $x = a^k$ for $k \leq m + 2n - 2$, the x-interval and the ax-interval are always split into three, the bx-interval is split into two, and there is a $\$x$-interval of size one. In general, we might not know whether the two subintervals of bx-interval are $bx\$$- and bxa-, or $bx\$$- and bxb-, or bxa- and bxb-intervals. However, since $x\$$- and $ax\$$-intervals both have size one, there can be no $bx\$$-interval, and thus all the subintervals can be uniquely determined and recursed on. When $x = a^{m+2n-1}$, the x-interval has size five and is split into three with the middle part (xa-interval) having size three. The ax interval has size three and is split into three. In this case too, only one combination of subintervals is possible.

When $x = a^{m+2n}$, the x-interval has size three and is split into three, and the $\$x$-, ax- and bx-intervals have size one. Therefore, the x-interval in the BWT contains some permutation of the three characters and all permutations are valid. This threeway permutation adds to the variation provided by the swaps in other parts of the BWT. A more careful analysis shows that the BWT x-interval of

- $\$ab$ or $\$ba$ implies an occurrence of $\$x\$$ which is only possible if $x\$$ is a separate string;
- $ba\$$ implies an occurrence of axa which is only possible if a single a is separate string;
- $a\$b$ implies occurrences of $ax\$$ and $\$xa$ which is only possible if $ax\$$ is a separate string;
- $ab\$$ implies an occurrence of $ax\$xb$; and
- $b\$a$ implies an occurrence of $bx\$xa$.

A single string solution is only possible in the last two cases, and any such solution corresponds to a solution for the BCSILA instance LCP_W (obtained by replacing $ax\$x$ or $x\$ax$ with x). Hence $\mathrm{LCP}_{W_\$}$ is a yes-instance of CSILA, and thus of BTSILA, if and only if LCP_W is a yes-instance of BCSILA, which proves the following result.

▶ **Theorem 16.** *BTSILA is NP-complete.*

8 Algorithm for CSSILA

In all of the above, we have assumed a binary alphabet (excluding the single symbol \$). In this section, we consider the CSSILA problem (i.e. Cyclic String Set Inference from LCP Array) without a restriction on the alphabet size (see [21] for more details).

Let $\mathcal{L}[1..n)$ be an instance of the CSSILA problem, i.e., an array of integers (and possibly ω's). Let $\sigma - 1$ be the number of zeroes in $\mathcal{L}$, and Σ an alphabet of size σ. As with the binary BCSSILA problem, we describe an algorithm that outputs a representation of the set $W_\mathcal{L} = \{w \in \Sigma^n : \mathrm{LCP}_{\mathrm{IBWT}(w)} = \mathcal{L}\}$; in this case the representation is an automaton that accepts $W_\mathcal{L}$. We show the following result.

▶ **Theorem 17.** *Given an array $\mathcal{L}[1..n)$ of integers (and possibly ω's) containing $\sigma-1$ zeroes, we can construct a deterministic finite automaton recognizing $W_{\mathcal{L}}$ in time $O(\sigma^2 2^\sigma(\frac{n}{\sigma}+1)^\sigma)$ and space $O(\sigma 2^\sigma(\frac{n}{\sigma}+1)^\sigma)$.*

References

1 Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004. doi:10.1016/S1570-8667(03)00065-0.

2 Amihood Amir. Personal communication, String Masters in Rouen, France, 3–5 February, 2014.

3 Alberto Apostolico. The myriad virtues of subword trees. In *Combinatorial Algorithms on Words*, NATO Advanced Science Institutes Series F12, pages 85–96. Springer-Verlag, 1985. doi:10.1007/978-3-642-82456-2_6.

4 Alberto Apostolico, Maxime Crochemore, Martin Farach-Colton, Zvi Galil, and S. Muthukrishnan. 40 years of suffix trees. *Communications of the ACM*, 59(4):66–73, 2016. doi:10.1145/2810036.

5 Hideo Bannai, Shunsuke Inenaga, Ayumi Shinohara, and Masayuki Takeda. Inferring strings from graphs and arrays. In *Proceedings of Mathematical Foundations of Computer Science 2003*, volume 2747 of *Lecture Notes in Computer Science*, pages 208–217. Springer, 2003. doi:10.1007/978-3-540-45138-9_15.

6 Bastien Cazaux and Eric Rivals. Reverse engineering of compact suffix trees and links: A novel algorithm. *Journal of Discrete Algorithms*, 28:9–22, 2014. doi:10.1016/j.jda.2014.07.002.

7 Julien Clément, Maxime Crochemore, and Giuseppina Rindone. Reverse engineering prefix tables. In *Proceedings of 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 289–300, 2009. doi:10.4230/LIPIcs.STACS.2009.1825.

8 Maxime Crochemore and Lucian Ilie. Computing longest previous factor in linear time and applications. *Information Processing Letters*, 106(2):75–80, 2008. doi:10.1016/j.ipl.2007.10.006.

9 Maxime Crochemore, Costas S. Iliopoulos, Solon P. Pissis, and German Tischler. Cover array string reconstruction. In *Proceeding of 21st Annual Symposium on Combinatorial Pattern Matching, CPM 2010*, volume 6129 of *Lecture Notes in Computer Science*, pages 251–259. Springer, 2010. doi:10.1007/978-3-642-13509-5_23.

10 Jean-Pierre Duval, Thierry Lecroq, and Arnaud Lefebvre. Border array on bounded alphabet. *Journal of Automata, Languages and Combinatorics*, 10(1):51–60, 2005.

11 Jean-Pierre Duval, Thierry Lecroq, and Arnaud Lefebvre. Efficient validation and construction of border arrays and validation of string matching automata. *RAIRO Theoretical Informatics and Applications*, 43(2):281–297, 2009. doi:10.1051/ita:2008030.

12 Frantisěk Franěk, Shudi Gao, Weilin Lu, Patrick J. Ryan, William F. Smyth, Yu Sun, and Lu Yang. Verifying a border array in linear time. *Journal on Combinatorial Mathematics and Combinatorial Computing*, 42:223–236, 2002. doi:10.1.1.32.5012.

13 Pawel Gawrychowski, Artur Jez, and Lukasz Jez. Validating the Knuth-Morris-Pratt failure function, fast and online. *Theory of Computing Systems*, 54(2):337–372, 2014. doi:10.1007/s00224-013-9522-8.

14 Ira M. Gessel and Christophe Reutenauer. Counting permutations with given cycle structure and descent set. *Journal of Combinatorial Theory, Series A*, 64(2):189–215, 1993. doi:10.1016/0097-3165(93)90095-P.

15 Dan Gusfield. *Algorithms on Strings, Trees, and Sequences : Computer Science and Computational Biology*. Cambridge University Press, Cambridge, United Kingdom, 1997. doi:10.1017/CBO9780511574931.

16 Jing He, Hongyu Liang, and Guang Yang. Reversing longest previous factor tables is hard. In *Proceedings of 12th International Symposium on Algorithms and Data Structures, WADS 2011*, volume 6844 of *Lecture Notes in Computer Science*, pages 488–499. Springer, 2011. doi:10.1007/978-3-642-22300-6_41.

17 Peter M. Higgins. Burrows-Wheeler transformations and de Bruijn words. *Theoretical Computer Science*, 457:128–136, 2012. doi:10.1016/j.tcs.2012.07.019.

18 Tomohiro I, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Verifying and enumerating parameterized border arrays. *Theoretical Computer Science*, 412(50):6959–6981, 2011. doi:10.1016/j.tcs.2011.09.008.

19 Tomohiro I, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Inferring strings from suffix trees and links on a binary alphabet. *Discrete Applied Mathematics*, 163:316–325, 2014. doi:10.1016/j.dam.2013.02.033.

20 Juha Kärkkäinen, Dominik Kempa, and Marcin Piątkowski. Tighter bounds for the sum of irreducible LCP values. *Theoretical Computer Science*, 656:265–278, 2015. doi:10.1016/j.tcs.2015.12.009.

21 Juha Kärkkäinen, Marcin Piatkowski, and Simon J. Puglisi. String inference from the LCP array. *CoRR*, abs/1606.04573, 2016. URL: http://arxiv.org/abs/1606.04573.

22 Gregory Kucherov, Lilla Tóthmérész, and Stéphane Vialette. On the combinatorics of suffix arrays. *Information Processing Letters*, 113(22-24):915–920, 2013. doi:10.1016/j.ipl.2013.09.009.

23 Udi Manber and Gene W. Myers. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993. doi:10.1137/0222058.

24 Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, and Marinella Sciortino. An extension of the Burrows-Wheeler transform. *Theoretical Computer Science*, 387(3):298–312, 2007. doi:10.1016/j.tcs.2007.07.014.

25 Yuto Nakashima, Takashi Okabe, Tomohiro I, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Inferring strings from Lyndon factorization. In *Proceedings of Mathematical Foundations of Computer Science 2014, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 565–576. Springer, 2014. doi:10.1007/978-3-662-44465-8_48.

26 Enno Ohlebusch. *Bioinformatics Algorithms: Sequence Analysis, Genome Rearrangements, and Phylogenetic Reconstruction*. Oldenbusch Verlag, 2013.

27 Nicolas Philippe. Caractérisation et énumération des arbres compacts des suffixes. Master's thesis, Université de Rouen, 2007.

28 Klaus-Bernd Schürmann and Jens Stoye. Counting suffix arrays and strings. *Theoretical Computer Science*, 395(2-3):220–234, 2008. doi:10.1016/j.tcs.2008.01.011.

29 Imre Simon. Piecewise testable events. In *Proceedings of 2nd GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975. doi:10.1007/3-540-07407-4_23.

30 Bill Smyth. *Computing Patterns in Strings*. Pearson Addison-Wesley, Essex, England, 2003.

31 Tatiana A. Starikovskaya and Hjalte Wedel Vildhøj. A suffix tree or not a suffix tree? *Journal of Discrete Algorithms*, 32:14–23, 2015. doi:10.1016/j.jda.2015.01.005.

32 Peter Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th Annual Symposium on Switching and Automata Theory 1973*, pages 1–11. IEEE Computer Society, 1973. doi:10.1109/SWAT.1973.13.

Neighborhood Complexity and Kernelization for Nowhere Dense Classes of Graphs*†

Kord Eickmeyer[1], Archontia C. Giannopoulou[2], Stephan Kreutzer[3], O-joung Kwon[4], MichałPilipczuk[5], Roman Rabinovich[6], and Sebastian Siebertz[7]

1 **Technische Universität Darmstadt, Darmstadt, Germany**
eickmeyer@mathematik.tu-darmstadt.de
2 **Technische Universität Berlin, Berlin, Germany**
archontia.giannopoulou@tu-berlin.de
3 **Technische Universität Berlin, Berlin, Germany**
stephan.kreutzer@tu-berlin.de
4 **Technische Universität Berlin, Berlin, Germany**
o-joung.kwon@tu-berlin.de
5 **University of Warsaw, Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl
6 **Technische Universität Berlin, Berlin, Germany**
roman.rabinovich@tu-berlin.de
7 **University of Warsaw, Warsaw, Poland**
siebertz@mimuw.edu.pl

Abstract

We prove that whenever G is a graph from a nowhere dense graph class $\mathcal{C}$, and A is a subset of vertices of G, then the number of subsets of A that are realized as intersections of A with r-neighborhoods of vertices of G is at most $f(r,\varepsilon)\cdot|A|^{1+\varepsilon}$, where r is any positive integer, ε is any positive real, and f is a function that depends only on the class $\mathcal{C}$. This yields a characterization of nowhere dense classes of graphs in terms of *neighborhood complexity*, which answers a question posed by Reidl et al. [26]. As an algorithmic application of the above result, we show that for every fixed integer r, the parameterized Distance-r Dominating Set problem admits an almost linear kernel on any nowhere dense graph class. This proves a conjecture posed by Drange et al. [9], and shows that the limit of parameterized tractability of Distance-r Dominating Set on subgraph-closed graph classes lies exactly on the boundary between nowhere denseness and somewhere denseness.

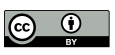
1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Graph Structure Theory, Nowhere Dense Graphs, Parameterized Complexity, Kernelization, Dominating Set

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.63

* The full version of this paper can be found as an arxiv preprint [11], https://arxiv.org/abs/1612.08197.

† The authors from Technische Universität Berlin (AG, SK, OK, and RR) have been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (ERC consolidator grant DISTRUCT, agreement No. 648527). The work of the authors from University of Warsaw (MP and Seb. S) is supported by the National Science Centre of Poland via POLONEZ grant agreement UMO-2015/19/P/ST6/03998. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 665778. M. Pilipczuk is supported by Foundation for Polish Science (FNP) via the START stipend programme.



44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 63; pp. 63:1–63:14

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

1 Introduction

Sparse graphs. The notion of nowhere denseness was introduced by Nešetřil and Ossona de Mendez [23, 24] as a general model of *uniform sparseness* of graphs. Many familiar classes of sparse graphs, like planar graphs, graphs of bounded treewidth, graphs of bounded degree, and, in fact, all classes that exclude a fixed (topological) minor, are nowhere dense. Notably, classes of bounded average degree or bounded degeneracy are not necessarily nowhere dense. In an algorithmic context this is reasonable, as every graph can be turned into a graph of degeneracy at most 2 by subdividing every edge once; however, the structure of the graph is essentially preserved under this operation.

▶ **Definition 1.** A *minor model* of a graph H in G is a family $(I_u)_{u\in V(H)}$ of pairwise vertex-disjoint connected subgraphs of G such that whenever $\{u,v\}$ is an edge in H, there are $u' \in I_u$ and $v' \in I_v$ for which $\{u',v'\}$ is an edge in G. The graph H is a *depth-r minor* of G, denoted $H \preccurlyeq_r G$, if there is a minor model $(I_u)_{u\in V(H)}$ of H in G such that each subgraph I_u has radius at most r.

▶ **Definition 2.** A class $\mathcal{C}$ of graphs is *nowhere dense* if there is a function $t : \mathbb{N} \to \mathbb{N}$ such that $K_{t(r)} \not\preccurlyeq_r G$ for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$.

Nowhere denseness turns out to be a very robust concept with several seemingly unrelated natural characterizations. These include characterizations by the density of shallow (topological) minors [23, 24], quasi-wideness [24] (a notion introduced by Dawar [6] in his study of homomorphism preservation properties), low tree-depth colorings [20], generalized coloring numbers [29], sparse neighborhood covers [16, 17], by a game called the splitter game [17] and by the model-theoretic concepts of stability and independence [1]. For a broader discussion we refer to the book of Nešetřil and Ossona de Mendez [25].

An important and related concept is the notion of a graph class of *bounded expansion* [20, 21, 22]. Precisely, a class of graphs $\mathcal{C}$ has bounded expansion if for any $r \in \mathbb{N}$, the ratio between the numbers of edges and vertices in any r-shallow minor of a graph from $\mathcal{C}$ is bounded by a constant depending on r only. Obviously, every class of bounded expansion is also nowhere dense, but the converse is not always true.

Domination problems. In the parameterized Dominating Set problem we are given a graph G and an integer parameter k, and the task is to determine the existence of a subset $D \subseteq V(G)$ of size at most k such that every vertex u of G is *dominated* by D, that is, u either belongs to D or has a neighbor in D. More generally, for fixed $r \in \mathbb{N}$ we can consider the Distance-r Dominating Set problem, where we are asked to determine the existence of a subset $D \subseteq V(G)$ of size at most k such that every vertex $u \in V(G)$ is within distance at most r from a vertex from D. The Dominating Set problem plays a central role in the theory of parameterized complexity, as it is a prime example of a W[2]-complete problem again with k as the paramenter, thus considered intractable in full generality from the parameterized point of view. For this reason, Dominating Set and Distance-r Dominating Set have been extensively studied in restricted graph classes, including the sparse setting.

The study of parameterized algorithms for Dominating Set on sparse and topologically constrained graph classes has a long history, and, arguably, it played a pivotal role in the development of modern parameterized complexity. A point of view that was particularly fruitful, and most relevant to our work, is *kernelization*. Recall that a kernelization algorithm is a polynomial-time preprocessing algorithm that transforms a given instance into an

equivalent one whose size is bounded by a function of the parameter only, independently of the overall input size. We are mostly interested in kernelization algorithms whose output guarantees are polynomial in the parameter, or maybe even linear. For Dominating Set on topologically restricted graph classes, linear kernels were given for planar graphs [2], bounded genus graphs [3], apex-minor-free graphs [12], graphs excluding a fixed minor [13], and graphs excluding a fixed topological minor [14]. All these results relied on applying tools of topological nature, most importantly deep decomposition theorems for graphs excluding a fixed (topological) minor. Notably, the research on kernelization for Dominating Set directly led to the introduction of the technique of *meta-kernelization* [3], which applies to a much larger family of problems on bounded-genus and H-minor-free graph classes.

Dawar and Kreutzer [7] showed that for every $r \in \mathbb{N}$ and every nowhere dense class $\mathcal{C}$, Distance-r Dominating Set is fixed-parameter tractable on $\mathcal{C}$. As far as polynomial kernelization is concerned, Drange et al. [9] gave a linear kernel for Distance-r Dominating Set on any graph class of bounded expansion[1], for every $r \in \mathbb{N}$, and an almost linear kernel for Dominating Set on any nowhere dense graph class; that is, a kernel of size $f(\varepsilon) \cdot k^{1+\varepsilon}$ for some function f. Drange et al. could not extend their techniques to larger domination radii r on nowhere dense classes, however, they conjectured that this should be possible. An important step was made recently by a subset of the authors [18], who gave a polynomial kernel for Distance-r Dominating Set on any nowhere dense class $\mathcal{C}$.

Nowhere dense classes are the limit for the fixed-parameter tractability of the problem: Drange et al. [9] showed that whenever $\mathcal{C}$ is a somewhere dense class closed under taking subgraphs, there is some $r \in \mathbb{N}$ for which Distance-r Dominating Set is $\mathsf{W}[2]$-hard on $\mathcal{C}$.

Neighborhood complexity. One of the crucial ideas in the work of Drange et al. [9] was to focus on the *neighborhood complexity* in sparse graph classes. For an integer $r \in \mathbb{N}$, a graph G, and a subset $A \subseteq V(G)$ of vertices of G, the *r-neighborhood complexity* of A, denoted $\nu_r(G, A)$, is defined as the number of different subsets of A that are of the form $N_r[u] \cap A$ for some vertex u of G; here, $N_r^G[u]$ denotes the ball of radius r around u. That is,

$$\nu_r(G, A) = |\{N_r^G[u] \cap A \ : \ u \in V(G)\}|.$$

It was proved by Reidl et al. [26] that linear neighborhood complexity exactly characterizes subgraph-closed classes of bounded expansion. More precisely, a subgraph-closed class $\mathcal{C}$ has bounded expansion if and only if for each $r \in \mathbb{N}$ there is a constant c_r such that $\nu_r(G, A) \leq c_r \cdot |A|$ for all graphs $G \in \mathcal{C}$ and vertex subsets $A \subseteq V(G)$. They posed as an open problem whether nowhere denseness can be similarly characterized by almost linear neighborhood complexity. The lack of a neighborhood complexity theorem for nowhere dense classes was a major, however not the only, obstacle preventing Drange et al. [9] from extending their kernelization results to Distance-r Dominating Set on any nowhere dense class. The neighborhood complexity result for nowhere dense classes for $r = 1$ was given by Gajarský et al. [15], and this result was used by Drange et al. [9] in their kernelization algorithm for Dominating Set (distance $r = 1$) on nowhere dense graph classes.

Conversely, if $\mathcal{C}$ is somewhere dense and closed under taking subgraphs, then for some $r \in \mathbb{N}$ it contains the exact r-subdivision of every graph [24]. In this case it is easy to see that the r-neighborhood complexity of a vertex subset A can be as large as $2^{|A|}$.

[1] Precisely, the kernelization algorithm of Drange et al. [9] outputs an instance of an annotated problem where some vertices are not required to be dominated; this will be the case in this paper as well.

Our results. In this paper we resolve in the affirmative both outlined conjectures. Let us first focus on neighborhood complexity.

▶ **Theorem 3.** *Let $\mathcal{C}$ be a graph class closed under taking subgraphs. Then $\mathcal{C}$ is nowhere dense if and only if there exists a function $f_{\mathrm{nei}}(r,\varepsilon)$ such that $\nu_r(G,A) \leq f_{\mathrm{nei}}(r,\varepsilon)\cdot|A|^{1+\varepsilon}$ for all $r\in\mathbb{N}$, $\varepsilon>0$, $G\in\mathcal{C}$, and $A\subseteq V(G)$.*

To prove the above, we carefully analyze the argument of Reidl et al. [26] for linear neighborhood complexity in classes of bounded expansion. This argument is based on the analysis of vertex orderings certifying the constant upper bound on the *weak coloring number* of any graph from a fixed class of bounded expansion. In the nowhere dense setting, we only have an n^{ε} upper bound on the weak coloring number, and therefore the reasoning breaks whenever one tries to use a bound that is exponential in this number. We circumvent this issue by applying tools based on model-theoretic properties of nowhere dense classes of graphs. More precisely, we use the fact that every nowhere dense class is *stable* in the sense of Shelah, and hence every graph H which is obtained from a graph G from a nowhere dense class via a first-order interpretation has bounded VC-dimension [1]. Then exponential blow-ups can be reduced to polynomial using the Sauer-Shelah Lemma. These tools were recently used by a subset of the authors to give polynomial bounds for uniform quasi-wideness [18], a fact that also turns out to be useful in our proof.

We remark that we were informed by Micek, Ossona de Mendez, Oum, and Wood [19] that they have independently proved the statement of Theorem 3 using different methods.

Having the almost linear neighborhood complexity for any nowhere dense class of graphs, we can revisit the argumentation of Drange et al. [9] and prove the following result.

▶ **Theorem 4.** *Let $\mathcal{C}$ be a fixed nowhere dense class of graphs, let r be a fixed positive integer, and let $\varepsilon>0$ be any fixed real. Then there is a polynomial-time algorithm that, given a graph $G\in\mathcal{C}$ and a positive integer k, returns a subgraph $G'\subseteq G$ and a vertex subset $Z\subseteq V(G')$ with the following properties:*

- *there is a set $D\subseteq V(G)$ of size at most k which r-dominates G if and only if there is a set $D'\subseteq V(G')$ of size at most k which r-dominates Z in G'; and*
- *$|V(G')|\leq f_{\mathrm{ker}}(r,\varepsilon)\cdot k^{1+\varepsilon}$, for some function $f_{\mathrm{ker}}(r,\varepsilon)$ depending only on the class $\mathcal{C}$.*

Just as in Drange et al. [9], the obtained triple (G',Z,k) is formally not an instance of Distance-r Dominating Set, but of an annotated variant of this problem where some vertices (precisely $V(G')\setminus Z$) are not required to be dominated. This is an annoying formal detail, however it can be addressed almost exactly as in Drange et al. by additional gadgeteering of annotations; see the full version for details.

Our proof of Theorem 4 revisits the line of reasoning of Drange et al. [9] for bounded expansion classes, and improves it in several places where the arguments could not be immediately lifted to the nowhere dense setting. The key ingredient is, of course, the usage of the newly proven almost linear neighborhood complexity for nowhere dense classes (Theorem 3), however, this was not the only piece missing. Another issue was that the algorithm of Drange et al. starts by iteratively applying the constant-factor approximation algorithm for Distance-r Dominating Set of Dvořák [10] in order to expose a certain structure in the instance. This part does not carry over to the nowhere dense setting, but we are able to circumvent it by using the new polynomial bounds for uniform quasi-wideness [18].

All proofs which are omitted in this extended abstract are marked with ($\star$). The full version of the paper can be found as an arxiv preprint [11].

2 Preliminaries

We use standard graph notation; see e.g. [8] for reference. All graphs considered in this paper are finite, simple, and undirected. For a graph G, by $V(G)$ and $E(G)$ we denote the vertex and edge sets of G, respectively.

Nowhere denseness (see Definition 2) admits several equivalent definitions, see [25] for a wider discussion. We next recall the ones used in this paper, as well as related concepts.

Weak coloring numbers. For a graph G we let $\Pi(G)$ denote the set of all linear orders of $V(G)$. For $L \in \Pi(G)$, $u, v \in V(G)$, and any $r \geq 0$, we say that u is *weakly r-reachable* from v with respect to L, if there is a path P of length at most r connecting u and v such that u is the smallest among the vertices of P with respect to L. By $\mathrm{WReach}_r[G, L, v]$ we denote the set of vertices that are weakly r-reachable from v with respect to L. For any subset $A \subseteq V(G)$, we let $\mathrm{WReach}_r[G, L, A] = \bigcup_{v \in A} \mathrm{WReach}_r[G, L, v]$. The *weak r-coloring number* $\mathrm{wcol}_r(G)$ of G is defined as

$$\mathrm{wcol}_r(G) \quad = \quad \min_{L \in \Pi(G)} \max_{v \in V(G)} \big|\mathrm{WReach}_r[G, L, v]\big|.$$

As proved by Zhu [29], the weak coloring numbers can be used to characterize bounded expansion and nowhere dense classes of graphs.

▶ **Theorem 5** ([29]). *Let $\mathcal{C}$ be a nowhere dense class of graphs. There is a function $f_{\mathrm{wcol}}(r, \varepsilon)$ such that $\mathrm{wcol}_r(H) \leq f_{\mathrm{wcol}}(r, \varepsilon) \cdot |V(H)|^{\varepsilon}$ for every $r \in \mathbb{N}$, $\varepsilon > 0$, and $H \subseteq G \in \mathcal{C}$.*

Quasi-wideness. A set $B \subseteq V(G)$ is called *r-independent* in G if for all distinct $u, v \in B$ we have $\mathrm{dist}_G(u, v) > r$.

▶ **Definition 6.** A class $\mathcal{C}$ of graphs is *uniformly quasi-wide* if there are functions $N \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ such that for all $r, m \in \mathbb{N}$ and all subsets $A \subseteq V(G)$ for $G \in \mathcal{C}$ of size $|A| \geq N(r, m)$ there is a set $S \subseteq V(G)$ of size $|S| \leq s(r)$ and a set $B \subseteq A \setminus S$ of size $|B| \geq m$ which is r-independent in $G - S$. The functions N and s are called the *margins* of the class $\mathcal{C}$.

It was shown by Nešetřil and Ossona de Mendez [24] that a class $\mathcal{C}$ of graphs is nowhere dense if and only if it is uniformly quasi-wide. For us it will be important that the margins N and s can be assumed to be polynomial in r and that the sets B and S can be efficiently computed. This was proved only recently by Kreutzer et al. [18].

▶ **Theorem 7** ([18]). *Let $\mathcal{C}$ be a nowhere dense class of graphs and let $t \colon \mathbb{N} \to \mathbb{N}$ be a function such that $K_{t(r)} \not\preccurlyeq_r G$ for all $r \in \mathbb{N}$ and all $G \in \mathcal{C}$. For every $r \in \mathbb{N}$ there exist constants $p(r)$ and $s(r) \leq t(r)$ such that for all $m \in \mathbb{N}$, all $G \in \mathcal{C}$, and all sets $A \subseteq V(G)$ of size at least $m^{p(r)}$, there is a set $S \subseteq V(G)$ of size at most $s(r)$ such that there is a set $B \subseteq A \setminus S$ of size at least m which is r-independent in $G - S$. Furthermore, there is an algorithm that, given an n-vertex graph $G \in \mathcal{C}$, $\varepsilon > 0$, $r \in \mathbb{N}$, and $A \subseteq V(G)$ of size at least $m^{p(r)}$, computes sets S and $B \subseteq A$ as described above in time $\mathcal{O}(r \cdot t \cdot |A|^{t+1} \cdot n^{1+\varepsilon})$.*

We remark that the running time of the algorithm of Theorem 7 is stated in the SODA version [18] only as $\mathcal{O}(r \cdot t \cdot n^{t+6})$. A finer analysis with the running times as stated above can be found in the arXiv version of that paper.

VC-dimension. Let $\mathcal{F} \subseteq 2^A$ be a family of subsets of a set A. For a set $X \subseteq A$, we denote $X \cap \mathcal{F} = \{X \cap F : F \in \mathcal{F}\}$. The set X is *shattered by* $\mathcal{F}$ if $X \cap \mathcal{F} = 2^X$. The *Vapnik-Chervonenkis dimension*, short *VC-dimension*, of $\mathcal{F}$ is the maximum size of a set X that is shattered by $\mathcal{F}$. Note that if X is shattered by $\mathcal{F}$, then also every subset of X is shattered by $\mathcal{F}$.

The following theorem was first proved by Vapnik and Chervonenkis [5], and rediscovered by Sauer [27] and Shelah [28]. It is often called the Sauer-Shelah Lemma in the literature.

▶ **Theorem 8** (Sauer-Shelah Lemma). *If $|A| \leq n$ and $\mathcal{F} \subseteq 2^A$ has VC-dimension d, then $|\mathcal{F}| \leq \sum_{i=0}^{d} \binom{n}{i} \in \mathcal{O}(n^d)$.*

Note that in the interesting cases $d \geq 2, n \geq 2$ it holds that $\sum_{i=0}^{d} \binom{n}{i} \leq n^d$, and in general it holds that $\sum_{i=0}^{d} \binom{n}{i} \leq 2 \cdot n^d$. For a graph G, the VC-dimension of G is defined as the VC-dimension of the family $\{N[v] \colon v \in V(G)\}$ of sets over the set $V(G)$.

VC-dimension and nowhere denseness. Adler and Adler [1] have proved that any nowhere dense class $\mathcal{C}$ of graphs is *stable*, which in particular implies that any class of structures obtained from $\mathcal{C}$ by means of a first-order interpretation has VC-dimension bounded by a constant depending only on $\mathcal{C}$ and the interpretation. In particular, the following is an immediate corollary of the results of Adler and Adler [1].

▶ **Corollary 9.** *Let $\mathcal{C}$ be a nowhere dense class of graphs and let $r \in \mathbb{N}$. For $G \in \mathcal{C}$, let $G_{=r}$ be the graph with the same vertex set as G and an edge $\{u, v\} \in E(G_{=r})$ if and only if $\mathrm{dist}_G(u, v) = r$. Define the graph $G_{\leq r}$ in the same manner, but putting an edge $\{u, v\}$ into $E(G_{\leq r})$ if and only if $\mathrm{dist}_G(u, v) \leq r$. Then there is an integer $c(r)$ such that both $G_{=r}$ and $G_{\leq r}$ have VC-dimension at most $c(r)$ for every $G \in \mathcal{C}$.*

By combining Corollary 9 with the Sauer-Shelah Lemma we infer the following.

▶ **Corollary 10.** *Let $\mathcal{C}$ be a nowhere dense class of graphs and $r \in \mathbb{N}$. Then $\nu_r(G, A) \leq |A|^{c(r)}$ for every graph $G \in \mathcal{C}$ and $A \subseteq V(G)$, where $c(r)$ is the constant given by Corollary 9.*

Thus, a polynomial bound on the neighborhood complexity for any nowhere dense class, and, in fact, for any stable class, already follows from known tools. Our goal in the next section will be to show that with the assumption of nowhere denseness we can prove an almost linear bound, as described in Theorem 3.

Distance profiles. In our reasoning we will need a somewhat finer view of the neighborhood complexity. More precisely, we would like to partition the vertices of the graph not only with respect to their r-neighborhood in a fixed set A, but also with respect to what are the exact distances of the elements of this r-neighborhood from the considered vertex. With this intuition in mind, we introduce the notion of a *distance profile*.

Let G be a graph and let $A \subseteq V(G)$ be a subset of its vertices. For a vertex $u \in V(G)$, the *r-distance profile* of u, denoted $\pi_r^G[u, A]$, is a function mapping vertices of A to $\{0, 1, \ldots, r, \infty\}$ defined as follows:

$$\pi_r^G[u, A](v) = \begin{cases} \mathrm{dist}_G(u, v) & \text{if } \mathrm{dist}_G(u, v) \leq r, \\ \infty & \text{otherwise.} \end{cases}$$

We say that a function $f \colon A \to \{0, 1, \ldots, r, \infty\}$ is *realized as an r-distance profile on A* if there is $u \in V(G)$ such that $f = \pi_r^G[u, A]$. We may drop the superscript if the graph is clear from the context.

Similarly to the neighborhood complexity, we define the *distance profile complexity* of a vertex subset $A \subseteq V(G)$ in a graph G, denoted $\widehat{\nu}_r(G, A)$, as the number of different functions realized as r-distance profiles on A in G. Clearly it always holds that $\nu_r(G, A) \leq \widehat{\nu}_r(G, A)$, thus Theorem 3 will follow directly from the following result, which will be proved in the next section.

▶ **Theorem 11.** *Let $\mathcal{C}$ be a nowhere dense class of graphs. Then there is a function $f_{\mathrm{nei}}(r, \varepsilon)$ such that for every $r \in \mathbb{N}$, $\varepsilon > 0$, graph $G \in \mathcal{C}$, and vertex subset $A \subseteq V(G)$, it holds that $\widehat{\nu}_r(G, A) \leq f_{\mathrm{nei}}(r, \varepsilon) \cdot |A|^{1+\varepsilon}$.*

Let us observe that the polynomial bound of Corollary 10 carries over to distance profiles.

▶ **Lemma 12 (⋆).** *Let $\mathcal{C}$ be a nowhere dense class of graphs. Then there is an integer $d(r)$ such that for every $r \in \mathbb{N}$, $\varepsilon > 0$, graph $G \in \mathcal{C}$, and vertex subset $A \subseteq V(G)$ with $|A| \geq 2$, it holds that $\widehat{\nu}_r(G, A) \leq |A|^{d(r)}$.*

3 Neighborhood complexity of nowhere dense classes

In this section we prove Theorem 11, which directly implies Theorem 3, as explained in the previous section. Our approach is to carefully analyze the proof of Reidl et al. [26] for bounded expansion classes, and to fix parts that break down in the nowhere dense setting using tools derived, essentially, from the stability of nowhere dense classes.

We first prove the following auxiliary lemma. We believe it may be of independent interest, as it seems very useful for the analysis of weak coloring numbers in the nowhere dense setting.

▶ **Lemma 13.** *Let $\mathcal{C}$ be a nowhere dense class of graphs and let $G \in \mathcal{C}$. For $r \geq 0$ and a linear order $L \in \Pi(G)$, let*

$$\mathcal{W}_{r,L} = \{\mathrm{WReach}_r[G, L, v] \,:\, v \in V(G)\}.$$

Then there is a constant $x(r)$, depending only on $\mathcal{C}$ and r (and not on G and L), such that $\mathcal{W}_{r,L}$ has VC-dimension at most $x(r)$.

Proof. Since $\mathcal{C}$ is nowhere dense, according to Theorem 5, it is uniformly quasi-wide, say with margins N and s. We fix a number m to be determined later, depending only on r and $\mathcal{C}$. Let $x = x(r) = N(2r, m)$ and $s = s(r)$. Assume towards a contradiction that there is a set $A \subseteq V(G)$ of size x which is shattered by $\mathcal{W}_{r,L}$. Fix sets $S \subseteq V(G)$ and $B \subseteq A \setminus S$ such that $|B| = m$, $|S| \leq s$, and B is $2r$-independent in $G - S$. We will treat L also as a linear order on the vertex set of $G - S$.

As a subset of A, the set B is also shattered by $\mathcal{W}_{r,L}$. That is, for every $X \subseteq B$ there is a vertex $v_X \in V(G)$ such that $X = \mathrm{WReach}_r[G, L, v_X] \cap B$. Note that since B is $2r$-independent in $G - S$, so is X, and we have $|\mathrm{WReach}_r[G - S, L, v_X] \cap B| \leq 1$.

For a vertex $\sigma \in S$, number $\rho \in \{0, \ldots, r-1\}$, and vertex $v \in V(G)$, let us consider the set $\mathcal{P}_{v,\sigma,\rho}$ of all paths of length (exactly) ρ that connect σ and v. We define $b_{\sigma,\rho}(v)$ to be the largest (with respect to L) vertex $b \in B$, for which there exists a path $P \in \mathcal{P}_{v,\sigma,\rho}$ such that every vertex on P is strictly larger than b with respect to L. If no such vertex in B exists, we put $b_{\sigma,\rho}(v) = \bot$. The *signature* of a vertex $v \in V(G)$ is defined as

$$\chi(v) = \big(b_{\sigma,\rho}(v)\big)_{\sigma \in S,\, 0 \leq \rho \leq r-1}.$$

It now follows that the number of possible signatures is small.

▶ **Claim 14** (⋆). *The set* $\{\chi(v)\colon v \in V(G)\}$ *has size at most* $(m+1)^{r\cdot s}$.

The next claim intuitively shows that for a vertex v, the signature of v plus the set of vertices of B that are weakly reachable from v in $G - S$ provide enough information to deduce precisely the set of vertices of B that are weakly reachable from v in G.

▶ **Claim 15** (⋆). *Suppose* $v, w \in V(G)$ *are such that*

$$\chi(v) = \chi(w) \quad \text{and} \quad \mathrm{WReach}_r[G-S, L, v] \cap B = \mathrm{WReach}_r[G-S, L, w] \cap B.$$

Then $\mathrm{WReach}_r[G, L, v] \cap B = \mathrm{WReach}_r[G, L, w] \cap B$.

By Claim 14 there are at most $(m+1)^{r\cdot s}$ possible signatures, while we argued that the intersection $\mathrm{WReach}_r[G-S, L, v] \cap B$ is always of size at most 1, hence there are at most $(m+1)$ possibilities for it. Thus, by Claim 15 we conclude that only at most $(m+1)^{r\cdot s}\cdot(m+1)$ subsets of B are realized as $B \cap W$ for some $W \in \mathcal{W}_{r,L}$. To obtain a contradiction with B being shattered by $\mathcal{W}_{r,L}$, it suffices to select m so that $(m+1)^{r\cdot s+1} < 2^m$. Since $s = s(r)$ is a constant depending on $\mathcal{C}$ and r only, we may choose m depending on $\mathcal{C}$ and r so that the above inequality holds. ◀

With Lemma 13 in hand, we now are ready to prove Theorem 11.

Proof of Theorem 11. Fix $r \geq 1$ and $\varepsilon > 0$. We also use small constants $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4 > 0$, which will be determined in the course of the proof.

The first step is to reduce the problem to the case when the size of the graph is bounded polynomially in $|A|$. Without loss of generality assume $|A| \geq 2$. According to Lemma 12, there is an integer $d(r)$ such that there are at most $|A|^{d(r)}$ different r-distance profiles on A. We therefore classify the elements of $V(G)$ according to their r-distance profiles on A, that is, we define an equivalence relation $\sim$ on $V(G)$ as follows:

$$v \sim w \quad \text{if and only if} \quad \pi_r^G[v, A] = \pi_r^G[w, A].$$

Construct a set A' by taking A and, for each equivalence class κ of $\sim$, adding an arbitrary element v_κ to A'. Then, construct a set A'' by starting with A', and, for each distinct $u, v \in A''$, performing the following operation: if $\mathrm{dist}_G(u, v) \leq r$, then add the vertex set of any shortest path between u and v to A''. Finally, let $G' = G[A'']$.

▶ **Claim 16** (⋆). *It holds that* $|A''| \leq |A|^{2\cdot d(r)+3}$.

▶ **Claim 17** (⋆). *It holds that* $\widehat{\nu}_r(G', A) \geq \widehat{\nu}_r(G, A)$.

The gain from this step is that the size of G' is bounded polynomially in terms of $|A|$, hence we can use better bounds on the weak coloring numbers, as explained next.

According to Theorem 5, there is a function f_{wcol} such that

$$\mathrm{wcol}_{2r}(G') \leq f_{\mathrm{wcol}}(2r, \varepsilon_4)\cdot |A''|^{\varepsilon_4} = f_{\mathrm{wcol}}(2r, \varepsilon_4)\cdot |A|^{(2\cdot d(r)+3)\cdot\varepsilon_4} = f_{\mathrm{wcol}}(2r, \varepsilon_4)\cdot |A|^{\varepsilon_3},$$

where $\varepsilon_4 = \varepsilon_3/(2\cdot d(r)+3)$. Let L be a linear order of $V(G')$ with $|\mathrm{WReach}_{2r}[G', L, v]| \leq f_{\mathrm{wcol}}(2r, \varepsilon_4)\cdot |A|^{\varepsilon_3}$. For each $v \in V(G')$, let us define the following set:

$$Y[v] = \mathrm{WReach}_r[G', L, v] \cap \mathrm{WReach}_r[G', L, A].$$

In other words, $Y[v]$ comprises all vertices that are weakly r-reachable both from v and from some vertex of A. Since $Y[v] \subseteq \mathrm{WReach}_r[G', L, v]$ for each $v \in V(G')$, we have $|Y[v]| \leq |\mathrm{WReach}_r[G', L, v]| \leq f_{\mathrm{wcol}}(2r, \varepsilon_4) \cdot |A|^{\varepsilon_3}$. Furthermore, as for each $v \in V(G')$ we have $Y[v] \subseteq \mathrm{WReach}_r[G, L, A] = \bigcup_{w \in A} \mathrm{WReach}_r[G, L, w]$, we have $\left|\bigcup_{v \in V(G')} Y[v]\right| \leq |A| \cdot \max_{w \in V(G')} |\mathrm{WReach}_r[G, L, w]| \leq f_{\mathrm{wcol}}(2r, \varepsilon_4) \cdot |A|^{1+\varepsilon_3}$.

We now classify the vertices $v \in V(G')$ according to their distance profiles $\pi_r^{G'}[v, Y[v]]$. More precisely, let $\equiv$ be the equivalence relation on $V(G')$ defined as follows:

$$v \equiv w \quad \text{if and only if} \quad Y[v] = Y[w] \text{ and } \pi_r^{G'}[v, Y[v]] = \pi_r^{G'}[w, Y[w]].$$

We next show that the equivalence relation $\equiv$ refines the standard partitioning according to r-distance profiles on A.

▶ **Claim 18** ($\star$). *For every $v, w \in V(G)$, if $v \equiv w$, then $\pi_r^{G'}[v, A] = \pi_r^{G'}[w, A]$.*

Claim 18 suggests the following approach to bounding $\widehat{\nu}_r(G', A)$: first give an upper bound on the number of possible sets of the form $Y[v]$, and then for each such set, bound the number of r-distance profiles on it. We deal with the second part first, as it essentially follows from Lemma 12. Let us set $\varepsilon_2 = \varepsilon_3 \cdot d(r)$, where $d(r)$ is the constant given by Lemma 12.

▶ **Claim 19** ($\star$). *There is $g(r, \varepsilon_4)$ such that for all $v \in V(G)$, we have $\widehat{\nu}_r(G, Y[v]) \leq g(r, \varepsilon_4)|A|^{\varepsilon_2}$.*

It remains to give an upper bound on the number of distinct sets $Y[v]$. Let us set $\varepsilon_1 = \varepsilon_3 \cdot x(r)$, where $x(r)$ is the constant given by Lemma 13.

▶ **Claim 20.** *It holds that $|\{Y[v] \colon v \in V(G')\}| \leq 1 + 2 \cdot f_{\mathrm{wcol}}(2r, \varepsilon_4)^{x(r)+1} \cdot |A|^{1+\varepsilon_1+\varepsilon_3}$.*

Proof. Let $\mathcal{Y} = \{Y[v] \colon v \in V(G')\} \setminus \{\emptyset\}$ be the family of all non-empty sets of the form $Y[v]$ for $v \in V(G')$; it suffices to show that $|\mathcal{Y}| \leq f_{\mathrm{wcol}}(2r, \varepsilon_4)^{x(r)+1} \cdot |A|^{1+\varepsilon_1+\varepsilon_3}$. Define mapping $\gamma \colon \mathcal{Y} \to V(G')$ as follows: for $Z \in \mathcal{Y}$, $\gamma(Z)$ is the largest element of Z with respect to L.

Take any $v \in V(G')$ with $Y[v] \neq \emptyset$, and recall that every vertex in $Y[v]$ is weakly r-reachable from v. Observe that every vertex $w \in Y[v]$ is weakly $2r$-reachable from $\gamma(Y[v])$. To see this, concatenate the two paths of length at most r that certify that $w \in \mathrm{WReach}_r[G', L, v]$ and $\gamma(Y[v]) \in \mathrm{WReach}_r[G', L, v]$, and note that this path of length at most $2r$ certifies that $w \in \mathrm{WReach}_{2r}[G', L, \gamma(Y[v])]$. Consequently, for every $y \in \gamma(\mathcal{Y})$, we have

$$\bigcup \gamma^{-1}(y) \subseteq \mathrm{WReach}_{2r}[G', L, y].$$

Hence the union of all sets of $\mathcal{Y}$ that choose the same y via γ has size at most $\mathrm{wcol}_{2r}(G')$.

Fix any $y \in \gamma(\mathcal{Y})$ and denote $S_y = \bigcup \gamma^{-1}(y)$. Let us count how many distinct subsets of S_y belong to $\mathcal{Y}$. Every such $Z = Y[v]$, as a subset of S_y, satisfies $Y[v] \cap S_y = Y[v] = \mathrm{WReach}_r[G'L, v] \cap \mathrm{WReach}_r[G', L, A]$. As for all v the set $\mathrm{WReach}_r[G', L, A]$ is the same, this means that the number of different $Y[v] \in \mathcal{Y}$ that are mapped to a fixed y is not larger than the number of different sets $\mathrm{WReach}_r[G', L, v] \cap S_y$, for $v \in V(G')$.

By Lemma 13, the set $\mathcal{W}_{r,L} = \{\mathrm{WReach}_r[G', L, v] \colon v \in V(G')\}$ has VC-dimension at most $x(r)$, and so has the subfamily $\{S_y \cap \mathrm{WReach}_r[G', L, v] \colon v \in V(G')\}$. Hence, by the Sauer-Shelah Lemma, we infer that $|\{S_y \cap \mathrm{WReach}_r[G', L, v] \colon v \in V(G')\}| \leq 2 \cdot |S_y|^{x(r)}$.

Finally, we observe that $\gamma(\mathcal{Y}) \subseteq \bigcup \mathcal{Y}$ and recall that $|\bigcup \mathcal{Y}| \leq f_{\mathrm{wcol}}(2r, \varepsilon_4) \cdot |A|^{1+\varepsilon_3}$, hence

$$\begin{aligned}
|\mathcal{Y}| &\leq \sum_{y \in \gamma(\mathcal{Y})} |\{S_y \cap \mathrm{WReach}_r[G', L, v] \colon v \in V(G')\}| \\
&\leq 2 \cdot \sum_{y \in \gamma(\mathcal{Y})} |S_y|^{x(r)} \leq 2 \cdot |\gamma(\mathcal{Y})| \cdot (\mathrm{wcol}_{2r}(G'))^{x(r)} \\
&\leq 2 \cdot |\bigcup \mathcal{Y}| \cdot (f_{\mathrm{wcol}}(2r, \varepsilon_4) \cdot |A|^{\varepsilon_3})^{x(r)} \\
&\leq 2 \cdot f_{\mathrm{wcol}}(2r, \varepsilon_4) \cdot |A|^{1+\varepsilon_3} \cdot f_{\mathrm{wcol}}(2r, \varepsilon_4)^{x(r)+1} \cdot |A|^{\varepsilon_1} \\
&\leq 2 \cdot f_{\mathrm{wcol}}(2r, \varepsilon_4)^{x(r)+1} \cdot |A|^{1+\varepsilon_1+\varepsilon_3}.
\end{aligned}$$

◀

By combining Claim 17, Claim 18, Claim 19, and Claim 20, we conclude that

$$\begin{aligned}
\widehat{\nu}_r(G, A) &\leq \widehat{\nu}_r(G', A) \leq \mathrm{index}(\equiv) \\
&\leq |\{Y[v] \ : \ v \in V(G')\}| \cdot g(r, \varepsilon_4) \cdot |A|^{\varepsilon_2} \\
&\leq (1 + 2 \cdot f_{\mathrm{wcol}}(2r, \varepsilon_4)^{x(r)+1} \cdot |A|^{1+\varepsilon_1+\varepsilon_3}) \cdot g(r, \varepsilon_4) \cdot |A|^{\varepsilon_2} \\
&\leq 3 \cdot f_{\mathrm{wcol}}(2r, \varepsilon_4)^{x(r)+1} \cdot g(r, \varepsilon_4) \cdot |A|^{1+\varepsilon_1+\varepsilon_2+\varepsilon_3}.
\end{aligned}$$

Now fix $\varepsilon_4 > 0$ so that $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 < \varepsilon$ and set $f_{\mathrm{nei}}(r, \varepsilon) = 3 \cdot f_{\mathrm{wcol}}(2r, \varepsilon_4)^{x(r)+1} \cdot g(r, \varepsilon_4)$. ◀

4 Kernelization for distance-r dominating sets

In this section we use the neighborhood complexity tools to prove Theorem 4. Throughout the section we fix a nowhere dense class $\mathcal{C}$. Whenever we say that the running time of some algorithm on a graph G is *polynomial*, we mean that it is of the form $\mathcal{O}((|V(G)| + |E(G)|)^\alpha)$, where α is a universal constant that is independent of $\mathcal{C}$, r, ε, or any other constants defined in the context. However, the constants hidden in the $\mathcal{O}(\cdot)$-notation may depend on $\mathcal{C}$, r, and ε.

Projections and projection profiles. Let $G \in \mathcal{C}$ be a graph and let $A \subseteq V(G)$ be a subset of vertices. For vertices $v \in A$ and $u \in V(G) \setminus A$, a path P connecting u and v is called *A-avoiding* if none of its vertices apart from v belong to A. For a positive integer r, the *r-projection* of any $u \in V(G) \setminus A$ on A, denoted $M_r^G(u, A)$ is the set of all vertices $v \in A$ that can be connected to u by an A-avoiding path of length at most r. The *r-projection profile* of a vertex $u \in V(G) \setminus A$ on A is a function $\rho_r^G[u, A]$ mapping vertices of A to $\{0, 1, \ldots, r, \infty\}$, defined as follows: for every $v \in A$, the value $\rho_r^G[u, A](v)$ is the length of a shortest A-avoiding path connecting u and v, and ∞ in case this length is larger than r. Similarly as for r-neighborhoods and r-distance profiles, we define

$$\mu_r(G, A) = |\{M_r^G(u, A) \colon u \in V(G) \setminus A\}| \quad \text{and} \quad \widehat{\mu}_r(G, A) = |\{\rho_r^G[u, A] \colon u \in V(G) \setminus A\}|$$

to be the number of different r-projections and r-projection profiles realized on A, respectively. Clearly, again it always holds that $\mu_r(G, A) \leq \widehat{\mu}_r(G, A)$. The following lemma is a simple consequence of the results of the previous section.

▶ **Lemma 21** (⋆). *Suppose $\mathcal{C}$ is a nowhere dense class of graphs. Then there is a function $f_{\mathrm{proj}}(r, \varepsilon)$ such that for every $r \in \mathbb{N}$, $\varepsilon > 0$, graph $G \in \mathcal{C}$, and vertex subset $A \subseteq V(G)$, it holds that $\widehat{\mu}_r(G, A) \leq f_{\mathrm{proj}}(r, \varepsilon) \cdot |A|^{1+\varepsilon}$.*

We next recall the main tool for projections proved by Drange et al. [9], namely the *Closure Lemma*. Intuitively, it says that any vertex subset $A \subseteq V(G)$ can be "closed" to a set $\mathrm{cl}_r(A)$ that is not much larger than A, such that all r-projections on $\mathrm{cl}_r(A)$ are small. The next lemma follows from a straightforward adaptation of the proof of Drange et al.

▶ **Lemma 22** (⋆, Lemma 2.9 of [9], adjusted)**.** *There is a function $f_{\mathrm{cl}}(r,\varepsilon)$ and a polynomial-time algorithm that, given $G \in \mathcal{C}$, $X \subseteq V(G)$, $r \in \mathbb{N}$, and $\varepsilon > 0$, computes the* r-closure *of X, denoted $\mathrm{cl}_r(X)$ with the following properties.*

- $X \subseteq \mathrm{cl}_r(X) \subseteq V(G)$*;*
- $|\mathrm{cl}_r(X)| \leq f_{\mathrm{cl}}(r,\varepsilon) \cdot |X|^{1+\varepsilon}$*; and*
- $|M_r^G(u, \mathrm{cl}_r(X))| \leq f_{\mathrm{cl}}(r,\varepsilon) \cdot |X|^{\varepsilon}$ *for each $u \in V(G) \setminus \mathrm{cl}_r(X)$.*

We need another lemma from Drange et al., called the *Short Paths Closure Lemma*.

▶ **Lemma 23** (⋆, Lemma 2.11 of [9], adjusted)**.** *There is a function $f_{\mathrm{pth}}(r,\varepsilon)$ and a polynomial-time algorithm which on input $G \in \mathcal{C}$, $X \subseteq V(G)$, $r \in \mathbb{N}$, and $\varepsilon > 0$, computes a superset $X' \supseteq X$ of vertices with the following properties:*

- *whenever* $\mathrm{dist}_G(u,v) \leq r$ *for $u, v \in X$, then* $\mathrm{dist}_{G[X']}(u,v) = \mathrm{dist}_G(u,v)$*; and*
- $|X'| \leq f_{\mathrm{pth}}(r,\varepsilon) \cdot |X|^{1+\varepsilon}$.

For the rest of this section let us fix constants $r \in \mathbb{N}$ and $\varepsilon > 0$; they will be used implicitly in the proofs. Let us recall some terminology from Drange et al. [9], which is essentially also present in the approach of Dawar and Kreutzer [7]. For a graph G, and vertex subset $Z \subseteq V(G)$, we say that a subset of vertices D is a (Z,r)-*dominator* if $Z \subseteq N_r^G(D)$. We write $\mathbf{ds}_r(G,Z)$ for the smallest (Z,r)-dominator in G and $\mathbf{ds}_r(G)$ for the smallest $(V(G),r)$-dominator in G. The crux of the approach of Drange et al. [9] is to perform kernelization in two phases: first compute a small r-*domination core*, and then reduce the size of the graph in one step.

▶ **Definition 24.** An r-*domination core* of G is a subset $Z \subseteq V(G)$ such that every minimum size (Z,r)-dominator is also a distance-r dominating set in G.

We shall prove the following analogue of Theorem 4.11 of [9].

▶ **Lemma 25.** *There exists a function $f_{\mathrm{core}}(r,\varepsilon)$ and a polynomial-time algorithm that, given a graph $G \in \mathcal{C}$ and integer $k \in \mathbb{N}$, either correctly concludes that G cannot be r-dominated by k vertices, or finds an r-domination core $Z \subseteq V(G)$ of G of size at most $f_{\mathrm{core}}(r,\varepsilon) \cdot k^{1+\varepsilon}$.*

Starting with $Z = V(G)$, which is clearly an r-domination core of G, we try to iteratively remove vertices from Z while preserving the property that Z is an r-domination core of G. More precisely, we show the following lemma, which is the analogue of Theorem 4.12 of [9] and of Lemma 11 of [7].

▶ **Lemma 26.** *There exists a function $f_{\mathrm{core}}(r,\varepsilon)$ and a polynomial-time algorithm that, given a graph $G \in \mathcal{C}$, an integer $k \in \mathbb{N}$, and an r-domination core Z of G with $|Z| > f_{\mathrm{core}}(r,\varepsilon) \cdot k^{1+\varepsilon}$, either correctly concludes that Z cannot be r-dominated by k vertices, or finds a vertex $z \in Z$ such that $Z \setminus \{z\}$ is still an r-domination core of G.*

Observe that Lemma 25 follows by applying Lemma 26 iteratively until the size of the core is reduced to at most $f_{\mathrm{core}}(r,\varepsilon) \cdot k^{1+\varepsilon}$. This iteration is performed at most n times leading to an additional factor n in the running time in Lemma 25.

The first step of the proof of Lemma 26 is to find a suitable approximation of a (Z,r)-dominator. For this, we can rely on the classic $\mathcal{O}(\log \mathsf{OPT})$-approximation of Brönnimann and Goodrich [4] for the Hitting Set problem in set families of bounded VC-dimension, as explained in the next lemma.

▶ **Lemma 27 (⋆).** *There is a function $f_{\mathrm{apx}}(r,\varepsilon)$ and a polynomial-time algorithm that, given $G \in \mathcal{C}$ and $Z \subseteq V(G)$, computes a (Z,r)-dominator of size at most $f_{\mathrm{apx}}(r,\varepsilon)\cdot k^{1+\varepsilon}$, where k is the size of a minimum r-dominating set of Z.*

We are now ready to prove Lemma 26.

Proof of Lemma 26. The function $f_{\mathrm{core}}(r,\varepsilon)$ will be defined in the course of the proof. For convenience, for now we assume that $|Z| > f_{\mathrm{core}}(r,\varepsilon)\cdot k^{1+C\varepsilon}$ for some large constant C, for at the end we will rescale ε accordingly.

We first apply Lemma 27, to either conclude that there is no (Z,r)-dominator of size at most k or to compute a (Z,r)-dominator X of size at most $f_{\mathrm{apx}}(r,\varepsilon)\cdot k^{1+\varepsilon}$. This application takes polynomial time. In the first case we can reject the instance, hence assume that we are in the second case.

We apply the algorithm of Lemma 22 to the set X and distance parameter $3r$, thus computing its closure $\mathrm{cl}_{3r}(X)$, henceforth denoted by X_{cl}. By Lemma 22, we have

$$|X_{\mathrm{cl}}| \quad\leq\quad f_{\mathrm{cl}}(3r,\varepsilon)\cdot|X|^{1+\varepsilon} \leq f_{\mathrm{cl}}(3r,\varepsilon)\cdot f_{\mathrm{apx}}(r,\varepsilon)^{1+\varepsilon}\cdot k^{1+3\varepsilon},$$

and, for every $u \in V(G)\setminus X_{\mathrm{cl}}$,

$$|M^G_{3r}(u,X_{\mathrm{cl}})| \quad\leq\quad f_{\mathrm{cl}}(3r,\varepsilon)\cdot|X|^{\varepsilon} \leq f_{\mathrm{cl}}(3r,\varepsilon)\cdot f_{\mathrm{apx}}(r,\varepsilon)^{\varepsilon}\cdot k^{3\varepsilon}.$$

We now classify the elements of $Z\setminus X_{\mathrm{cl}}$ according to their $3r$-projection profiles on X_{cl}. More precisely, let us define an equivalence relation $\sim$ on $Z\setminus X_{\mathrm{cl}}$ as follows:

$$u \sim v \quad\text{if and only if}\quad \rho^G_{3r}[u,X_{\mathrm{cl}}] = \rho^G_{3r}[v,X_{\mathrm{cl}}].$$

By Lemma 21, the number of equivalence classes of $\sim$ is bounded as follows:

$$\mathrm{index}(\sim) \quad\leq\quad f_{\mathrm{nei}}(3r,\varepsilon)\cdot|X_{\mathrm{cl}}|^{1+\varepsilon} \leq f_{\mathrm{nei}}(3r,\varepsilon)\cdot f_{\mathrm{cl}}(3r,\varepsilon)^{1+\varepsilon}\cdot f_{\mathrm{apx}}(r,\varepsilon)^{(1+\varepsilon)^2}\cdot k^{1+7\varepsilon}.$$

Note that the partition of $V(G)$ into the equivalence classes of $\sim$ can be computed in polynomial time, by just computing the $3r$-projection profile for each vertex using breadth-first search, and then comparing the profiles pairwise.

Let $t(r)$, $p(r)$, and $s(r)$ be the functions provided by Theorem 7 for the class $\mathcal{C}$. Denote $\alpha = f_{\mathrm{cl}}(3r,\varepsilon)\cdot f_{\mathrm{apx}}(r,\varepsilon)^{\varepsilon}\cdot k^{3\varepsilon} + s(2r) + 1$ and $\beta = \alpha\cdot(r+2)^{s(r)}+1$. From the above inequalities on $|X_{\mathrm{cl}}|$ and $\mathrm{index}(\sim)$, it follows that setting $C = 7+3\cdot p(2r)$, we can fix the function $f_{\mathrm{core}}(r,\varepsilon)$ so that the following inequality is always satisfied:

$$f_{\mathrm{core}}(r,\varepsilon)\cdot k^{1+C\varepsilon} \geq |X_{\mathrm{cl}}| + \mathrm{index}(\sim)\cdot\beta^{p(2r)}.$$

Since we assumed that $|Z| > f_{\mathrm{core}}(r,\varepsilon)\cdot k^{1+C\varepsilon}$, it follows that $|Z\setminus X_{\mathrm{cl}}| > \mathrm{index}(\sim)\cdot\beta^{p(2r)}$. Hence, by the pigeonhole principle, there exists an equivalence class κ of $\sim$ that contains more than $\beta^{p(2r)}$ vertices. By applying the algorithm of Theorem 7 to any subset of κ of size exactly $\beta^{p(2r)}$, we find sets $S\subseteq V(G)$ and $L\subseteq\kappa\setminus S$ such that $|S|\leq s(r)$, $|L|\geq\beta$, and L is $2r$-independent in $G-S$. This application takes time $\mathcal{O}(r\cdot t(2r)\cdot\beta^{p(2r)\cdot(t(r)+1)}\cdot|V(G)|^{1+\varepsilon})$. Provided ε satisfies $3\cdot p(2r)\cdot(t(2r)+1)\cdot\varepsilon<1$, which we can assume without loss of generality, we have that $\beta^{p(2r)\cdot(t(2r)+1)}\leq\mathcal{O}(k)$; here, the constants hidden in the $\mathcal{O}(\cdot)$-notation may depend on $\mathcal{C}$. Since we can assume that $k\leq|V(G)|$, this application of the algorithm of Theorem 7 takes then time $\mathcal{O}(|V(G)|^{2+\varepsilon})$, which is polynomial with the degree independent of $\mathcal{C}$.

We now classify the elements of L according to their r-distance profiles on S. Note that $|L|\leq\beta$ and that the number of r-distance profiles on S is bounded by $(r+2)^{|S|}\leq(r+2)^{s(r)}$.

Since $\beta > \alpha \cdot (r+2)^{s(r)}$, by the pigeonhole principle we infer that there is a subset $R \subseteq L$ of size $|R| > \alpha$ such that

$$\pi_r^G(v, S) = \pi_r^G(w, S) \quad \text{for all} \quad v, w \in R.$$

At this point the situation is almost exactly as in Lemma 3.8 of [9]. More precisely, every vertex of R is an irrelevant dominatee, i.e., it can be excluded from Z without spoiling the property that Z is a domination core. The proof of the following claim is based on an exchange argument.

▶ **Claim 28** (⋆). *The set Z' is also a domination core of G.*

Claim 28 ensures us that vertex z is an irrelevant dominatee that can be returned by the algorithm. Note that we were able to find z provided $|Z| > f_{\mathrm{core}}(r, \varepsilon) \cdot k^{1+C\varepsilon}$ for some constant C depending on $\mathcal{C}$ and r. Hence, we conclude the proof by rescaling ε to ε/C throughout the reasoning. ◀

Finally, having computed a suitably small domination core, we can construct a kernel. This part of reasoning, encapsulated in the following lemma, is exactly the same as in [9].

▶ **Lemma 29** (⋆). *There exists a function $f_{\mathrm{fin}}(r, \varepsilon)$ and a polynomial-time algorithm that, given a graph $G \in \mathcal{C}$ and an r-domination core $Z \subseteq V(G)$ of G, computes a graph G' with at most $f_{\mathrm{fin}}(r, \varepsilon) \cdot |Z|^{1+\varepsilon}$ vertices such that $Z \subseteq V(G')$ and $\mathbf{ds}_r(G', Z) = \mathbf{ds}_r(G, Z)$.*

Theorem 4 follows by combining Lemma 25 and LEmma 29, and rescaling ε by factor 3.

References

1. Hans Adler and Isolde Adler. Interpreting nowhere dense graph classes as a classical notion of model theory. *European Journal of Combinatorics*, 36:322–330, 2014.
2. Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.
3. Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *J. ACM*, 63(5):44:1–44:69, 2016.
4. Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.
5. A. Chervonenkis and V. Vapnik. Theory of uniform convergence of frequencies of events to their probabilities and problems of search for an optimal solution from empirical data. *Automation and Remote Control*, 32:207–217, 1971.
6. Anuj Dawar. Homomorphism preservation on quasi-wide classes. *J. Comput. Syst. Sci.*, 76(5):324–332, 2010.
7. Anuj Dawar and Stephan Kreutzer. Domination problems in nowhere-dense classes. In *FSTTCS 2009*, volume 4 of *LIPIcs*, pages 157–168. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2009.
8. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2012.
9. Pål Grønås Drange, Markus Sortland Dregi, Fedor V. Fomin, Stephan Kreutzer, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, Felix Reidl, Fernando Sánchez Villaamil, Saket Saurabh, Sebastian Siebertz, and Somnath Sikdar. Kernelization and sparseness: the case of Dominating Set. In *STACS 2016*, volume 47 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. See arxiv preprint 1411.4575 for full proofs.

10 Zdeněk Dvořák. Constant-factor approximation of the domination number in sparse graphs. *European Journal of Combinatorics*, 34(5):833–840, 2013.

11 Kord Eickmeyer, Archontia C. Giannopoulou, Stephan Kreutzer, O-joung Kwon, Michal Pilipczuk, Roman Rabinovich, and Sebastian Siebertz. Neighborhood complexity and kernelization for nowhere dense classes of graphs. *CoRR*, abs/1612.08197, 2016. URL: `https://arxiv.org/abs/1612.08197`.

12 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *SODA 2010*, pages 503–510. SIAM, 2010.

13 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Linear kernels for (Connected) Dominating Set on H-minor-free graphs. In *SODA 2012*, pages 82–93. SIAM, 2012.

14 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Linear kernels for (Connected) Dominating Set on graphs with excluded topological subgraphs. In *STACS 2013*, volume 20 of *LIPIcs*, pages 92–103. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013.

15 Jakub Gajarský, Petr Hliněný, Jan Obdržálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. Kernelization using structural parameters on sparse graph classes. *J. Comput. Syst. Sci.*, 84:219–242, 2017.

16 Martin Grohe, Stephan Kreutzer, Roman Rabinovich, Sebastian Siebertz, and Konstantinos Stavropoulos. Colouring and covering nowhere dense graphs. In *WG 2015*, pages 325–338, 2015.

17 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. In *STOC 2014*, pages 89–98. ACM, 2014.

18 Stephan Kreutzer, Roman Rabinovich, and Sebastian Siebertz. Polynomial kernels and wideness properties of nowhere dense graph classes. In *SODA 2017*, pages 1533–1545. SIAM, 2017. See arxiv preprint 1608.05637 for full proofs.

19 Piotr Micek, Patrice Ossona de Mendez, Sang-il Oum, and David R. Wood. Personal communication, 2016.

20 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion I. Decompositions. *European Journal of Combinatorics*, 29(3):760–776, 2008.

21 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion II. Algorithmic aspects. *European Journal of Combinatorics*, 29(3):777–791, 2008.

22 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion III. Restricted graph homomorphism dualities. *European Journal of Combinatorics*, 29(4):1012–1024, 2008.

23 Jaroslav Nešetřil and Patrice Ossona de Mendez. First order properties on nowhere dense structures. *The Journal of Symbolic Logic*, 75(03):868–887, 2010.

24 Jaroslav Nešetřil and Patrice Ossona de Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

25 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity – Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

26 Felix Reidl, Fernando Sánchez Villaamil, and Konstantinos Stavropoulos. Characterising bounded expansion by neighbourhood complexity. *CoRR*, abs/1603.09532, 2016.

27 Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

28 Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972.

29 Xuding Zhu. Colouring graphs with bounded generalized colouring number. *Discrete Mathematics*, 309(18):5562–5568, 2009.

Additive Spanners and Distance Oracles in Quadratic Time

Mathias Bæk Tejs Knudsen*

University of Copenhagen, Copenhagen, Denmark
mathias@tejs.dk

Abstract

Let G be an unweighted, undirected graph. An additive k-spanner of G is a subgraph H that approximates all distances between pairs of nodes up to an additive error of $+k$, that is, it satisfies $d_H(u,v) \leq d_G(u,v) + k$ for all nodes u, v, where d is the shortest path distance. We give a deterministic algorithm that constructs an additive $O(1)$-spanner with $O\big(n^{4/3}\big)$ edges in $O\big(n^2\big)$ time. This should be compared with the randomized Monte Carlo algorithm by Woodruff [ICALP 2010] giving an additive 6-spanner with $O\big(n^{4/3}\log^3 n\big)$ edges in expected time $O\big(n^2\log^2 n\big)$.

An (α,β)-approximate distance oracle for G is a data structure that supports the following distance queries between pairs of nodes in G. Given two nodes u, v it can in constant time compute a distance estimate $\tilde{d}$ that satisfies $d \leq \tilde{d} \leq \alpha d + \beta$ where d is the distance between u and v in G. Sommer [ICALP 2016] gave a randomized Monte Carlo $(2,1)$-distance oracle of size $O\big(n^{5/3}\operatorname{poly}\log n\big)$ in expected time $O\big(n^2\operatorname{poly}\log n\big)$. As an application of the additive $O(1)$-spanner we improve the construction by Sommer [ICALP 2016] and give a Las Vegas $(2,1)$-distance oracle of size $O\big(n^{5/3}\big)$ in time $O\big(n^2\big)$. This also implies an algorithm that in $O\big(n^2\big)$ time gives approximate distance for all pairs of nodes in G improving on the $O\big(n^2\log n\big)$ algorithm by Baswana and Kavitha [SICOMP 2010].

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases graph algorithms, data structures, additive spanners, distance oracles

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.64

1 Introduction

Let $G = (V, E)$ be an unweighted, undirected graph on n nodes and m edges. A subgraph H of G is an *additive k-spanner* if the following holds for every pair u, v of nodes in G:

$$d_H(u,v) \leq d_G(u,v) + k\,,$$

where $d_H(u,v)$ and $d_G(u,v)$ is the distance between u and v in H and G respectively. This paper will only consider additive spanners and not multiplicative or mixed spanners, so we will simply say that H is a *k-spanner* when we mean that H is an additive k-spanner.

In this paper we consider algorithms constructing k-spanners, and there are therefore three interesting parameters: The distortion k, the running time of the algorithm, and the size of the spanner created. Elkin and Peleg [19] showed how to construct 2-spanners with $O\big(n^{3/2}\big)$ edges in $O\big(n^{5/2}\big)$ time, and Baswana et al [9] gave an algorithm that constructs 6-spanners with $O\big(n^{4/3}\big)$ edges in $O\big(n^{2/3}m\big)$ time.

* Research partly supported by Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research under the Sapere Aude research career programme and by the FNU project AlgoDisc – Discrete Mathematics, Algorithms, and Data Structures.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 64; pp. 64:1–64:12

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

Table 1 A summary of the performance of selected algorithms that creates a k-spanner H from a graph on n nodes. It shows the additive distortion, k, and an upper bound on the number of edges in H as well as the running time of the algorithm that constructs H.

k	Number of Edges	Running Time	Comment	Reference
2	$O(n^{3/2})$	$O(n^{5/2})$	Deterministic	[19]
2	$O(n^{3/2}\log^{1/2} n)$	$O(n^2\log^2 n)$	Deterministic	[18]
2	$O(n^{3/2})$	$O(n^2)$	Deterministic	Theorem 5
6	$O(n^{4/3})$	$O(n^{2/3}m)$	Deterministic	[9]
6	$O(n^{4/3}\log^3 n)$	$O(n^2\log^2 n)$	Randomized Monte Carlo	[33]
8	$O(n^{4/3})$	$O(n^2)$	Deterministic	Theorem 9

The running time of these algorithms can be improved if we allow the k-spanners to be larger by a $\operatorname{poly}\log n$ factor. Dor, Halperin and Zwick [18] showed that we can construct 2-spanners with $O\left(n^{3/2}\log^{1/2} n\right)$ edges in $O(n^2\log^2 n)$ time, and Woodruff [33] gave an algorithm to construct 6-spanners with $O(n^{4/3}\log^3 n)$ edges in $O(n^2\log^2 n)$ time. The construction of Woodruff is furthermore randomized Monte Carlo. These results are summarized in Table 1.

These improvements to the running time fit into the following paradigm. For a fixed k the authors find algorithms that produce spanners that are *almost* as small as the best known construction of k-spanners and have *near*-quadratic running time. We reverse this way of looking at the problem. We are now trying to find algorithms that yield k-spanners that are exactly as small as the best known constructions for any $k = O(1)$, i.e. $O(n^{4/3})$, and at the same time we want the algorithm to run as fast as possible. All known algorithms for creating $O(1)$-spanners that have close to optimal size run in time $\Omega(n^2)$. [1] So a natural question is to ask if there exists a $k = O(1)$ and an algorithm that constructs a k-spanner with $O(n^{4/3})$ edges in $O(n^2)$ time. In fact Sommer [28] mentioned at his talk at ICALP 2016 that the main obstacle towards getting a better running time for constructing the distance oracle he presented is the lack of such an algorithm. In his case the distortion $k = O(1)$ is only factored into the running time and not the distortion of oracle. Therefore, it does not matter what k is as long as it is constant.

We show that it possible to attain this goal by giving an algorithm that constructs 8-spanners deterministically with $O(n^{4/3})$ edges in $O(n^2)$ time. Comparing this with the algorithm by Woodruff [33] this gets rid of the $\log^3 n$ factor on the number of edges and a factor of $\log^2 n$ in the running time. Furthermore, the algorithm is deterministic and not randomized Monte Carlo. The price of these improvements is that the distortion is larger than 6. We note that there are no lower bounds ruling out the possibility of a 4-spanner with $O(n^{4/3})$ edges. For the application to the distance oracle by Sommer [28], the distortion is unimportant as long as it is constant. We also show how to construct 2-spanners with $O(n^{3/2})$ edges in $O(n^2)$ time. For a comparison to previous work see Table 1.

Related work. Elkin and Peleg [19] showed that[2] any graph on n nodes has a 2-spanner with $O(n^{3/2})$ edges, Chechik [15] showed that it has a 4-spanner with $O\left(n^{7/5}\log^{1/5} n\right)$ edges, and Baswana et al [9] showed that it has a 6-spanner with $O(n^{4/3})$ edges. These results are

[1] For instance the algorithm by Baswana et al [9] gives a 6-spanner with $O(n^{4/3})$ edges and is therefore only interesting when $m = \Omega(n^{4/3})$, in which case the running time is $\Theta(n^{2/3}m) = \Omega(n^2)$.

[2] Aingworth et al [5] earlier showed the same result up to logarithmic factors on the size of the spanner.

Table 2 For a given k an upper bound of $f(n)$ is a proof that any graph on n nodes has a k-spanner with no more than $f(n)$ edges. A lower bound of $g(n)$ is a proof that there exists a graph on n nodes for which any k-spanner must have at least $g(n)$ edges.

k	Upper Bound	Lower Bound	Reference
2 & 3	$O\big(n^{3/2}\big)$	$\Omega\big(n^{3/2}\big)$	[19]/[31]
4 & 5	$O\big(n^{7/5}\log^{1/5} n\big)$	$\Omega\big(n^{4/3}\big)$	[15]/[11]
≥ 6	$O\big(n^{4/3}\big)$	$n^{4/3-o(1)}$	[9]/[1]

complemented by a negative result of Abboud and Bodwin [1]. A consequence of their result is that for any $k = O(1)$ there exists a graph on n nodes such that any k-spanner of this graph has at least $n^{4/3-o(1)}$ edges.

Another negative result comes from Erdős's girth conjecture [20]. It states that for any constant k there exists graphs with n nodes and $\Omega\big(n^{1+1/k}\big)$ edges where the girth is $2k+2$. This conjecture has been proved for $k = 2, 3, 5$ [31, 11]. In particular if the conjecture is true this implies that there exists graphs for which any $(2k-1)$-spanner must have at least $\Omega\big(n^{1+1/k}\big)$ edges. Woodruff [32] proved that whether the conjecture is true or not, there exists a graph on n nodes such that any $(2k-1)$-spanner of the graph has at least $\Omega\big(k^{-1}n^{1+1/k}\big)$ edges.

There are also upper and lower bounds when we allow the distortion k to depend on n, see [14, 13, 15, 22]. In this paper, however, we are only interested in the case where $k = O(1)$. The upper and lower bounds for $k = O(1)$ are summarized in Table 2.

Techniques. Previous algorithms that construct k-spanners in $\tilde{O}\big(n^2\big)$ time all relied on constructing a hitting set for some set of neighbourhoods. In [18] this is done deterministically via a dominating set algorithm, and in [33] this is done via sampling. This approach will inherently come with the cost of a $\operatorname{poly}\log n$ factor. Furthermore, in the construction of 6-spanners by Woodruff [33] the number of neighbourhoods that need to be hit is so large that it seems impossible with current techniques to modify the algorithm to be Las Vegas. To avoid this we instead use a clustering approach described in Section 2. The algorithm in Theorem 9 is obtained using this clustering and a careful modification of the path-buying algorithm of [9].

Approximate Distance Oracles and All Pairs Almost Shortest Paths. Given an undirected an unweighted graph G an (α, β)-approximate distance oracle for G is a data structure that supports the following query. Given two nodes u, v it can compute a distance estimate $\tilde{d}$ that satisfies $d \leq \tilde{d} \leq \alpha d + \beta$ where d is the distance between u and v in G. For work on approximate distance oracles see e.g. [2, 3, 4, 6, 7, 8, 10, 12, 16, 17, 23, 24, 26, 27, 29, 30, 34]. Sommer [28] gave a randomized Monte Carlo $(2,1)$-distance oracle that can be constructed in $O\big(n^2 \operatorname{poly}\log n\big)$ time, has size $O\big(n^{5/3}\operatorname{poly}\log n\big)$ and can answer queries in $O(1)$ time. We improve the construction time and the size to $O\big(n^2\big)$ and $O\big(n^{5/3}\big)$ respectively, and our construction is randomized Las Vegas. As a corollary we can compute an estimate $\tilde{d}(u,v)$ for all pairs of nodes in G satisfying $d_G(u,v) \leq \tilde{d}(u,v) \leq 2d_G(u,v)+1$ in time $O\big(n^2\big)$. This improves upon the $O\big(n^2\log n\big)$ algorithm by Baswana and Kavitha [8].

Preliminaries. For a graph G and two nodes u, v we denote the distance from u to v in G by $d_G(u,v)$. All graphs considered in this paper are unweighted, and unless otherwise specified they are undirected as well. For an undirected graph G and a node u the neighbourhood of u is the set of nodes adjacent to u and is denoted by $\Gamma_G(u)$.

Overview. In Section 2 we introduce the clustering we use when constructing the spanners. In Section 3 we show how to create an 8-spanner with $O(n^{4/3})$ edges in $O(n^2)$ time and thereby prove Theorem 9. In Section 4 we provide the details on how to give an improved $(2,1)$-distance oracle.

2 Clustering

Our construction of additive spanners uses clustering techniques, and we present our clustering framework below. Let $G = (V, E)$ be a graph with n vertices and m edges. We let t be a parameter that can depend on our needs. For a sequence $u_1, \ldots, u_\ell$ of nodes we define the clusters $C_i, i \in \{1, \ldots, \ell\}$ by

$$C_i = (\Gamma_G(u_i) \cup \{u_i\}) \setminus (C_1 \cup \ldots \cup C_{i-1}) \,.$$

Furthermore we also define graphs $G_0, G_1, \ldots, G_\ell$ in the following way. We let $G_0 = G$, and for $i > 0$ we let G_i be the subgraph of G defined in the following way. The nodes of G_i are the same as the nodes of G. An edge (u, v) from G is contained in G_i unless both endpoints u and v are are contained in $C_1 \cup \ldots \cup C_i$. From each node u_i we let T_i be a BFS tree in G_{i-1} rooted at u_i.

▶ **Definition 1.** A sequence $u_1, \ldots, u_\ell$ is called a *t-clustering* if the following requirements are satisfied.

- The node u_i maximizes the size of $(\Gamma_G(u_i) \cup \{u_i\}) \setminus (C_1 \cup \ldots \cup C_{i-1})$.
- Every cluster C_i contains at least t nodes.
- For every node v we have $|(\Gamma_G(v) \cup \{v\}) \setminus (C_1 \cup \ldots \cup C_\ell)| < t$.

We say that a node v is *clustered* if $v \in C_1 \cup \ldots \cup C_\ell$ and *unclustered* otherwise. We note that since every cluster C_i contains at least t nodes and the clusters are disjoint we have $\ell \leq \frac{n}{t}$.

▶ **Lemma 2.** *Let $u_1, \ldots, u_\ell$ be a t-clustering of a graph $G = (V, E)$. For every $i = 1, 2, \ldots, \ell$ the number of edges in G_{i-1} is at most $n\,|C_i|$. The number of edges in G_ℓ is less than nt.*

Proof. The set of edges in G_{i-1} can be written as

$$\{\{u, v\} \mid u \in V, v \in (\Gamma_G(u)) \setminus (C_1 \cup \ldots \cup C_{i-1})\} \,.$$

Therefore, the number of edges in G_{i-1} is bounded by

$$\sum_{v \in V} |(\Gamma_G(v)) \setminus (C_1 \cup \ldots \cup C_{i-1})| \,. \tag{1}$$

Since every term in (1) is bounded by $|C_i|$, we conclude that the number of edges in G_i is at most $n\,|C_i|$.

In the same manner we see that the number of edges in G_ℓ is bounded by the sum $\sum_{v \in V} |(\Gamma_G(v)) \setminus (C_1 \cup \ldots \cup C_\ell)|$, which is clearly less than nt. ◀

▶ **Lemma 3.** *Let $u_1, \ldots, u_\ell$ be a t-clustering of $G = (V, E)$ and let $u, v \in V$ be a pair of nodes. Assume that some shortest path from u to v in G is not contained in G_ℓ from Lemma 2. Then there exists an index $i \in \{1, 2, \ldots, \ell\}$ such that*

$$d_{T_i}(u_i, u) + d_{T_i}(u_i, v) \leq d_G(u, v) + 2 \,.$$

Proof. Consider a shortest path p from u to v that is not contained in G_ℓ and let w be a clustered node on p such that $w \in C_i$. We choose w such that i is smallest possible. By choosing i smallest possible p is contained in G_{i-1}. Furthermore since the distance from w to u_i is at most 1 we see that

$$d_{G_{i-1}}(u_i, u) + d_{G_{i-1}}(u_i, v) \leq d_{G_{i-1}}(w, u) + d_{G_{i-1}}(w, v) + 2 = d_G(u, v) + 2\,.$$

Since T_i is a is shortest path tree in G_{i-1} the conclusion follows. ◀

▶ **Lemma 4.** *Given a graph G and a parameter $t > 0$ we can construct a t-clustering $u_1, \ldots, u_\ell$, the corresponding BFS trees $T_1, \ldots, T_\ell$ and G_ℓ in $O(n^2)$ time.*

Proof. The algorithm will work by finding the nodes $u_1, \ldots, u_\ell$ consecutively, i.e. first u_1, then u_2 and so on. The algorithm will maintain a graph G'. In the beginning of the algorithm we have $G' = G_0$, and after we add u_i we will alter G' such that $G' = G_i$. The total cost of altering all G' will be $O(m) = O(n^2)$.

We find u_i by looking at all nodes in $G' = G_{i-1}$ and count the number of neighbours not in $C_1 \cup \ldots \cup C_{i-1}$. Since G_{i-1} has at most $n\,|C_i|$ edges this takes $O(n\,|C_i|)$ time. Then the algorithm finds a BFS tree from u_i in G_{i-1} in $O(n\,|C_i|)$ time. Hence the total time used by the algorithm is:

$$O\left(m + \sum_{i=1}^{\ell} n\,|C_i|\right) = O(n^2)\,.$$ ◀

3 Constructing $O(1)$-Spanners

In this section we present our construction of an 8-spanner with $O\big(n^{4/3}\big)$ edges in $O\big(n^2\big)$ time. As a warmup we show how we can use the clustering from Section 2 to give a 2-spanner with $O\big(n^{3/2}\big)$ edges in $O\big(n^2\big)$ time.

▶ **Theorem 5.** *There exists an algorithm that given a graph G with n nodes constructs a 2-spanner of G with $\leq 2n^{3/2}$ edges in $O\big(n^2\big)$ time.*

Proof. Let $t = \sqrt{n}$ and construct a t-clustering $u_1, \ldots, u_\ell$ with Lemma 4. Let $H = T_1 \cup \ldots \cup T_\ell \cup G_\ell$. The number of edges in H is at most $n\ell + nt \leq 2n\sqrt{n}$ by Lemma 2 and the fact that $\ell \leq \frac{n}{t}$.

Now we just need to prove that H is a 2-spanner. Let u, v be arbitrary nodes and let p be a shortest path from u to v in G. We wish to prove that

$$d_H(u, v) \leq d_G(u, v) + 2\,. \tag{2}$$

If p is contained in G_ℓ then (2) is obviously true. Otherwise there exists an index i such that $d_{T_i}(u, v) \leq d_G(u, v) + 2$ by Lemma 3, and (2) is true since $T_i \subset H$. ◀

Next we turn to showing how to create an 8-spanner H with $O\big(n^{4/3}\big)$ edges in $O\big(n^2\big)$ time. The idea is the following. We start by creating a t-clustering $u_1, \ldots, u_\ell$ with $t = n^{1/3}$ and $\ell \leq n^{2/3}$. Using the BFS trees $T_1, \ldots, T_\ell$ along with Lemma 3 we can then get an additive 2-approximation of $d_G(u_i, u_j)$ for all pairs of indices i, j, which we will call $\delta_{i,j}$. The calculation of the BFS trees in $O\big(n^2\big)$ time relies on an idea similar to one in [5]. The BFS trees also gives us a path from u_i to u_j that is at most 2 longer than the shortest path. If we add all these shortest paths to our spanner along with G_ℓ and the neighbours in C_i of each

u_i we will get a 6-spanner. Unfortunately, adding a path could require adding up to $\Omega(\ell)$ edges, and since there are ℓ^2 pairs we can only guarantee that the spanner has $O(\ell^3)$ edges, which is $O(n^2)$ if $\ell \approx n^{2/3}$. (We only need to add edges on the path that are not already in G_ℓ) Instead we use an argument similar to the path-buying argument from [9] and the construction from [21]. We add the path from u_i to u_j unless we can guarantee that there is an additive 2-approximation of this path in the spanner already. We do this by maintaining an upper bound $\Delta_{i,j}$ on the distance from u_i to u_j in the spanner H. We then argue that if we add a path with k edges not already in the spanner, then there are $\Omega(k)$ pairs $u_{i'}, u_{j'}$ for which the upper bound $\Delta_{i',j'}$ is improved. Then, this will imply that at most $O(\ell^2)$ edges are added giving an upper bound of $O(n^{4/3})$ on the number of edges in H.

After this informal discussion of the construction we turn to the details. The algorithm is given a graph $G = (V, E)$ with n nodes and m edges, and will return a spanner $H = (V, F)$. Initially $F = \emptyset$ and we will add edges to H so that H becomes a 8-spanner of G. The algorithm starts by creating a t-clustering $u_1, \ldots, u_\ell$ with $t = n^{1/3}$ using Lemma 4 in $O(n^2)$ time. Since $\ell \leq \frac{n}{t}$ we have $\ell \leq n^{2/3}$. Then we add edges from u_i to all nodes in $C_i \setminus \{u_i\}$ to H for all $i \in \{1, 2, \ldots, \ell\}$. We add at most n edges this way. Then we add all edges from G_ℓ to H. This adds at most $nt = n^{4/3}$ edges to H.

We give each node $u \in V$ a color $c(u) \in \{0, 1, 2, \ldots, \ell\}$. If u is unclustered then u has color $c(u) = 0$. Otherwise $c(u) = i$ where i is the unique index such that $u \in C_i$. For each pair of indices $i, j \in \{1, 2, \ldots, \ell\}$ we define $\delta_{i,j}$ by:

$$\delta_{i,j} = \min_{k \in \{1,2,\ldots,\ell\}} \{d_{T_k}(u_k, u_i) + d_{T_k}(u_k, u_j)\} \,. \tag{3}$$

We first note that for a choice of i, j we can calculate the right hand side of (3) in $O(\ell)$ time since we are taking the minimum over ℓ different values. So in $O(\ell^3)$ time the algorithm calculates $\delta_{i,j}$ for all pairs of indices i, j. Since $\ell \leq n^{2/3}$ this is within the $O(n^2)$ time bound. As a consequence of Lemma 3 we get that $\delta_{i,j}$ is a good approximation of $d_G(u_i, u_j)$, more precisely:

$$d_G(u_i, u_j) \leq \delta_{i,j} \leq d_G(u_i, u_j) + 2 \,. \tag{4}$$

We now define T'_i to be the tree obtained from T_i by contracting each edge in G_ℓ. Since an edge is contained in G_ℓ iff at least one of its endpoints is unclustered we can construct T'_i from T_i in $O(n)$ time. The algorithm does so for all $i \in \{1, 2, \ldots, \ell\}$ in $O(n\ell) = O(n^{5/3})$ time. We note that the shortest path between two nodes u, v in T'_i contains exactly the edges on the shortest path between u, v in T_i excluding the edges that are contained in G_ℓ.

The algorithm initializes $\Delta_{i,j} = \infty$ for all pairs of indices i, j with $i \neq j$ and let $\Delta_{i,i} = 0$ for all i. We will maintain that $\Delta_{i,j}$ is an upper bound on $d_H(u_i, u_j)$ throughout the algorithm. Now the algorithm goes through all pairs u_i, u_j and adds a path of length at most 2 longer than a shortest path between the nodes if needed. Specifically, we do the following:

Let L be an upper bound on the number of nodes of the path p from u_i to u_j in T'_k on line 6. Then Algorithm 1 can implemented in $O(\ell^3 + \ell^2 L)$ time. Hence we just need to prove that $L = O(\ell)$ in order to conclude that it can be implemented in $O(\ell^3) = O(n^2)$ time. This follows from the fact that p is an almost shortest path and the following reasoning. If p contained $> C\ell$ nodes for some sufficiently large constant C it would contain more than C nodes of the same color. Since nodes of the same color have distance at most 2 in G this would imply that there was a much shorter path from u to v in G contradicting (4) if C was chosen large enough. The details with $C = 5$ are given in the following lemma:

▶ **Lemma 6.** *The path p contains no nodes of color 0, and at most 5 nodes of each color $\neq 0$.*

Algorithm 1

```
1  For each pair of indices i,j ∈ {1,2,...,ℓ}:
2      For all k ∈ {1,2,...,ℓ}:
3          Set Δ_{i,j} := min{Δ_{i,j}, Δ_{i,k} + Δ_{k,j}}.
4      If Δ_{i,j} > δ_{i,j} + 2 do:
5          Find a k ∈ {1,2,...,ℓ} such that d_{T_k}(u_k,u_i) + d_{T_k}(u_k,u_j) = δ_{i,j}.
6          Find the path p from u_i to u_j in T'_k.
7          Add all edges from p to H.
8          Write p = (w_0, w_1, w_2, ..., w_{s-1}).
9          For all x ∈ {0,1,2,...,s-1}:
10             Set y := d_{T_k}(u_i, w_x).
11             Set Δ_{i,c(w_x)} := min{Δ_{i,c(w_x)}, y + 1}
12             Set Δ_{c(w_x),j} := min{Δ_{c(w_x),j}, (δ_{i,j} − y) + 1}
```

Proof. Obviously p does not contain a node with color 0, since all its incident edges would be contained in G_ℓ and hence not in T'_k. Now assume for the sake of contradiction that p contains 6 nodes of some color $r \neq 0$. When traversing p from u_i to u_j let α and β be the first and the last node of color r respectively. The distance from α to β when following p must be at least 5 by assumption. On the other hand α and β have distance at most 2 in G. So there exists a path in G from u_i to u_j that is at least 3 edges shorter than p. This contradicts (4). Hence the assumption was wrong and p contains at most 5 nodes of each color $\neq 0$. ◀

Since there are ℓ different colors $\neq 0$ the path p contains at most 5ℓ nodes and the running time of Algorithm 1 is $O(n^2)$. So now we just need to prove that H is an 8-spanner and that H has at most $O(n^{4/3})$ edges. We start by proving that H is an 8-spanner. Here we will utilize that the $\Delta_{i,j}$ is an upper bound on the distance from u_i to u_j in H. Furthermore, Algorithm 1 guarantees that $\Delta_{i,j} \leq \delta_{i,j} + 2$. Together with (4) this gives that

$$d_H(u_i, u_j) \leq d_G(u_i, u_j) + 4\,. \tag{5}$$

▶ **Lemma 7.** *The subgraph H of G is an additive 8-spanner of G.*

Proof. Assume for the sake of contradiction that H is not an additive 8-spanner and let u, v be a pair of nodes with shortest possible distance in G such that:

$$d_H(u, v) > d_G(u, v) + 8\,. \tag{6}$$

Say that $d_G(u, v) = D$ and let $p = (w_0, w_1, \ldots, w_D)$ be a shortest path from u to v in G where $w_0 = u$ and $w_D = v$. Since the pair (u, v) has the smallest possible distance in G such that (6) holds and $d_G(w_1, v) = D - 1$ we have $d_H(w_1, v) \leq (D - 1) + 8$. In particular the edge (u, w_1) is not in H as it would contradict (6). Hence u cannot be unclustered, as all the edges incident to an unclustered node is contained in G_ℓ and therefore H. With the same reasoning we conclude that v is clustered. Let the colors of u and v be i and j respectively. The distances from u and v to u_i and u_j respectively are at most 1. Combining this insight with (5) we get:

$$d_H(u, v) \leq d_H(u_i, u_j) + 2 \leq d_G(u_i, u_j) + 6 \leq d_G(u, v) + 8\,.$$

But this contradicts the assumption (6). Hence the assumption was wrong and H is an additive 8-spanner of G. ◀

Lastly, we need to prove that H contains no more than $O(n^{4/3})$ edges. Informally, we argue the following way. Whenever the $s-1$ edges of p are added to H on line 7 of Algorithm 1 there are $\Omega(s)$ different colors on p. For each color r on p we then argue that either $\Delta_{i,r}$ or $\Delta_{r,j}$ are made smaller on line 11 or 12 of Algorithm 1. Lastly, we argue that $\Delta_{i,j}$ can only be updated $O(1)$ times, and since there are $\ell^2 \leq n^{4/3}$ variables $\Delta_{i,j}$ this implies that Algorithm 1 only adds $O(n^{4/3})$ edges to H. This intuition is formalized in Lemma 8 bellow:

▶ **Lemma 8.** *Algorithm 1 adds no more than $25\ell^2$ edges to H.*

Proof. Say that the algorithm adds the edges from the path $p = (w_0, w_1, \ldots, w_{s-1})$ on line 7 of Algorithm 1 where $w_0 = u_i, w_{s-1} = u_j$. First we note that since $d_G(u_i, u_j) \geq \delta_{i,j} - 2$ by (4) we have that $d_G(u_i, w_x) \geq y - 2$ for every $x \in \{0, 1, \ldots, s-1\}$, where we consider y to be a function of x defined by $y = d_{T_k}(u_i, w_x)$ as on line 10. Now fix x and let $r = c(w_x)$. Then there is an edge between w_x and u_r and therefore $d_G(u_i, u_r) \geq y-3$, i.e. $y+1 \leq d_G(u_i, u_r)+4$. So if Algorithm 1 decreases $\Delta_{i,r}$ on line 11 we have $\Delta_{i,r} \leq d_G(u_i, u_r) + 4$ after it is decreased. Since $\Delta_{i,r}$ is an upper bound on $d_H(u_i, u_r)$ and therefore also an upper bound on $d_G(u_i, u_r)$ we see that $\Delta_{i,r}$ can be decreased at most 5 times for each choice of i, r. By symmetry we see that we can also decrease $\Delta_{r,j}$ on line 12 at most 5 times. Since there are ℓ^2 pairs of indices the algorithm can change the values of $\Delta_{i,r}$ or $\Delta_{r,j}$ on line 11 and 12 of Algorithm 1 at most $5\ell^2$ times.

Let r be a color on p. After the execution of lines 9-12 we have

$$\Delta_{i,r} + \Delta_{r,j} \leq \delta_{i,j} + 2\,.$$

Due to the execution of lines 2 and 3 this was not the case before. Hence either $\Delta_{i,r}$ or $\Delta_{r,j}$ were updated. By Lemma 6 there are at least $\frac{s}{5}$ colors on p, so if the algorithm adds A edges in total it makes at least $\frac{A}{5}$ updates of upper bounds $\Delta_{i,r}$ or $\Delta_{r,j}$. Since there can be at most $5\ell^2$ such updates we conclude that $\frac{A}{5} \leq 5\ell^2$ and that Algorithm 1 adds no more than $25\ell^2$ edges. ◀

To summarize, the algorithm presented in this section runs in $O(n^2)$ time and gives an additive 8-spanner with no more than $26n^{4/3} + n = O(n^{4/3})$ edges. We have made no attempt to optimize the constant in the O-notation. Hence we get:

▶ **Theorem 9.** *There exists an algorithm that given a graph G with n nodes constructs an 8-spanner of G with $O(n^{4/3})$ edges in $O(n^2)$ time.*

4 Distance Oracles

In the following we show how to modify the construction by Sommer [28] to obtain a $(2,1)$-distance oracle of size $O(n^{5/3})$ that can be constructed in expected $O(n^2)$ time.

Let G be a given graph, and H an 8-spanner of G constructed by Theorem 9. H is constructed in $O(n^2)$ time and has $O(n^{4/3})$ edges. During the construction we use only $O(n^{5/3})$ space.

Let $u_1, u_2, \ldots, u_\ell$ be a $n^{1/3}$-clustering of G. Using Lemma 4 we obtain $T_1, \ldots, T_\ell$ and G_ℓ in $O(n^2)$ time. For each node v we define four *portals* $p_1(v), p_2(v), p_3(v), p_4(v)$. We define $p_1(v) = u_i$, where u_i is chosen such that the distance between v and u_i in T_i is minimized. In case of ties we choose the node u_i with the lowest index i. The node $p_{j+1}(v)$ for $j = 1, 2, 3$ is chosen depending on $p_j(v)$. If $p_j(v) = u_1$ we let $p_{j+1}(v) = u_1$. Otherwise $p_j(v) = u_i$ for some index i. We let $p_{j+1}(v) = u_{i'}$ where $u_{i'}$ is chosen among $u_1, u_2, \ldots, u_{i-1}$ such that the

distance between $u_{i'}$ and v in $T_{i'}$ is minimized. In case of ties we choose the node $u_{i'}$ with the lowest index i'. The portals for all nodes can be found in $O(n^{5/3})$ time.

We will use the following lemma by Pătraşcu and Roditty [23] to construct a $(2,1)$-distance oracle for G_ℓ, that uses space $O(n^{5/3})$.

▶ **Lemma 10** ([23]). *For any unweighted, undirected graph, there exists a distance oracle of size $O(n^{5/3})$ that, given any nodes u and v at distance d, returns a distance of at most $2d+1$ in constant time. The distance oracle can be constructed in expected time $O(mn^{2/3})$.*

In the proof in [23] they only claim a running time of $O(mn^{2/3}+n^{7/3})$, however, this can be fixed to give the correct running time of $O(mn^{2/3})$ [25]. By [23, Claim 9] it is easy to see how to get a running time of $O(mn^{2/3}+n^2)$ which suffice for our purposes.

We are now ready to define the distance oracle. For each $i=1,2,\ldots,\ell$ we store the distances $d_{T_i}(u_i,v)$ and $d_H(u_i,v)$ for all nodes v. The distances $d_H(u_i,v)$ can be calculated using a BFS in time $O(\ell n^{4/3})=O(n^2)$. For each node v we store its portals $p_j(v), j=1,2,3,4$. We augment this distance oracle with the Pătraşcu-Roditty distance oracle from Lemma 10 for G_ℓ.

We now show how to use the distance oracle to obtain approximate distances for a query u,v. We let $\delta_{PR}(u,v)$ be the approximate distance in G_ℓ returned by the Pătraşcu-Roditty distance oracle. We define $\delta_j(u,v)$ in the following way. Let $p_j(u)=u_i$. Then $\delta_j(u,v)=d_{T_i}(u_i,u)+\min\{d_{T_i}(u_i,v),d_H(u_i,v)\}$. The distance returned by the distance oracle is the minimum of $\delta_{PR}(u,v)$, $\delta_j(u,v)$ and $\delta_j(v,u)$ for $j=1,2,3,4$.

We will now argue that if the the distance between u and v is d, then the distance oracle returns a distance between d and $2d+1$. The distance returned is obviously at least d, so we just need to show that it is at most $2d+1$. Consider a shortest path between u and v in G. If there is at most one node on the shortest path which is incident to a node u_i in the clustering then the shortest path is contained in G_ℓ, and therefore:

$$\delta_{PR}(u,v)\leq 2d_{G_\ell}(u,v)+1=2d+1\,.$$

So assume that there exists an edge on the shortest path not in G_ℓ. Let i be the smallest index such that there is an edge (z,t) on the shortest path with $z,t\in C_1\cup\ldots\cup C_i$. Say that z is closer to u than to v in G. Assume that $z\in C_i$ and $t\in C_{i'}$ for some index $i'\leq i$ (the case where $z\in C_{i'}$ and $t\in C_i$ is handled symmetrically). Since the shortest path is contained in G_{i-1} and $G_{i'-1}$ we have that

$$d_{T_i}(u_i,u)+d_{T_{i'}}(u_{i'},v)\leq(d_G(u,z)+1)+(d_G(t,v)+1)=d+1\,,$$

and therefore:

$$\min\{d_{T_i}(u_i,u),d_{T_i}(u_i,v)\}\leq\frac{d+1}{2}\,.$$

Assume that $d_{T_i}(u_i,u)\leq\frac{d+1}{2}$. The other case is handled similarly. Say that $p_j(u)=u_{k_j}$ for $j=1,2,3,4$. First assume that $k_j>i$ for all $j=1,2,3,4$. Then we conclude that $d_{T_{k_1}}(p_1(u),u)\leq d_{T_i}(u_i,u)-4$. The distance returned by the distance oracle is at most

$$\begin{aligned}\delta_1(u,v)&\leq d_{T_{k_1}}(p_1(u),u)+d_H(p_1(u),v)\\&\leq d_{T_{k_1}}(p_1(u),u)+d_G(p_1(u),v)+8\\&\leq 2d_{T_{k_1}}(p_1(u),u)+d_G(u,v)+8\\&\leq 2(d_{T_i}(u_i,u)-4)+d+8\leq 2d+1\,.\end{aligned}$$

Now assume that $k_j \leq i$ for some $j \in \{1,2,3,4\}$ and let j be the smallest index such that $k_j \leq i$. By definition we have that $d_{T_{k_j}}(p_j(u),u) \leq d_{T_i}(u_i,u)$. Furthermore the shortest path is contained in G_{k_j-1} and therefore $d_{T_{k_j}}(p_j(u),v) \leq d_{T_{k_j}}(p_j(u),u) + d_G(u,v)$. The distance returned is at most

$$\begin{aligned}\delta_j(u,v) &\leq d_{T_{k_j}}(p_j(u),u) + d_{T_{k_j}}(p_j(u),v)\\ &\leq 2d_{T_{k_j}}(p_j(u),u) + d\\ &\leq 2d_{T_i}(u_i,u) + d \leq 2d+1\,.\end{aligned}$$

We conclude that the distance returned by the distance oracle is always between d and $2d+1$. The result is summarized in Theorem 11.

▶ **Theorem 11.** *For any unweighted, undirected graph, there exists a distance oracle of size $O(n^{5/3})$ that, given any nodes u and v at distance d, returns a distance of at most $2d+1$ in constant time. The distance oracle can be constructed in expected time $O(n^2)$.*

Acknowledgements. The author would like to thank Christian Sommer for helpful discussions on the application of the 8-spanner to his construction of distance oracles.

References

1 Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. In *Proc. 48th ACM Symposium on Theory of Computing (STOC)*, pages 351–361, 2016.

2 Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with affine stretch. In *Distributed Computing – 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings*, pages 404–415, 2011. doi:10.1007/978-3-642-24100-0_39.

3 Rachit Agarwal. The space-stretch-time tradeoff in distance oracles. In *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 49–60, 2014. doi:10.1007/978-3-662-44777-2_5.

4 Rachit Agarwal and Philip Brighten Godfrey. Brief announcement: a simple stretch 2 distance oracle. In *ACM Symposium on Principles of Distributed Computing, PODC'13, Montreal, QC, Canada, July 22-24, 2013*, pages 110–112, 2013. doi:10.1145/2484239.2484277.

5 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999. See also SODA'96.

6 Surender Baswana, Akshay Gaur, Sandeep Sen, and Jayant Upadhyay. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 609–621, 2008. doi:10.1007/978-3-540-70575-8_50.

7 Surender Baswana, Vishrut Goyal, and Sandeep Sen. All-pairs nearly 2-approximate shortest paths in I time. *Theor. Comput. Sci.*, 410(1):84–93, 2009. doi:10.1016/j.tcs.2008.10.018.

8 Surender Baswana and Telikepalli Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM Journal on Computing*, 39(7):2865–2896, 2010.

9 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (alpha, beta)-spanners. *ACM Trans. Algorithms*, 7(1):5, 2010. See also SODA'05.

10 Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $o(n^2)$ time. *ACM Transactions on Algorithms (TALG)*, 2(4):557–577, 2006.

11 Clark T. Benson. Minimal regular graphs of girth eight and twelve. *Canad. J. Math*, 18(1):94, 1966.

12 Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *Workshop on Algorithms and Data Structures*, pages 541–552. Springer, 2007.

13 Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 855–872, 2016. doi:10.1137/1.9781611974331.ch61.

14 Gregory Bodwin and Virginia Vassilevska Williams. Very sparse additive spanners and emulators. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 377–382, 2015. doi:10.1145/2688073.2688103.

15 Shiri Chechik. New additive spanners. In *Proc. 24th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 498–512, 2013. doi:10.1137/1.9781611973105.36.

16 Shiri Chechik. Approximate distance oracles with constant query time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 654–663. ACM, 2014.

17 Shiri Chechik. Approximate distance oracles with improved bounds. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 1–10. ACM, 2015.

18 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000. See also FOCS'96.

19 Michael Elkin and David Peleg. $(1+\varepsilon, \beta)$-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004. See also STOC'01.

20 Paul Erdős. Extremal problems in graph theory. In *"Theory of Graphs and its Applications," Proc. Sympos. Smolenice*. Citeseer, 1964.

21 Mathias Bæk Tejs Knudsen. Additive spanners: A simple construction. In *Proc. 14th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 277–281, 2014.

22 Seth Pettie. Low distortion spanners. *ACM Trans. Algorithms*, 6(1):7:1–7:22, 2009. doi:10.1145/1644015.1644022.

23 Mihai Pătraşcu and Liam Roditty. Distance oracles beyond the thorup–zwick bound. *SIAM Journal on Computing*, 43(1):300–311, 2014.

24 Mihai Pătraşcu, Liam Roditty, and Mikkel Thorup. A new infinity of distance oracles for sparse graphs. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 738–747. IEEE, 2012.

25 Liam Roditty. personal communication.

26 Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *International Colloquium on Automata, Languages, and Programming*, pages 261–272. Springer, 2005.

27 Christian Sommer. Shortest-path queries in static networks. *ACM Computing Surveys (CSUR)*, 46(4):45, 2014.

28 Christian Sommer. All-pairs approximate shortest paths and distance oracle preprocessing. In *LIPIcs – Leibniz International Proceedings in Informatics*, volume 55. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

29 Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 703–712. IEEE, 2009.

30 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

31 Rephael Wenger. Extremal graphs with no C4's, C6's, or C10's. *Journal of Combinatorial Theory, Series B*, 52(1):113–116, 1991.

32 David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 389–398, 2006.

33 David P. Woodruff. Additive spanners in nearly quadratic time. In *Proc. 37th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 463–474, 2010.

34 Christian Wulff-Nilsen. Approximate distance oracles with improved preprocessing time. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 202–208. Society for Industrial and Applied Mathematics, 2012.

Finding, Hitting and Packing Cycles in Subexponential Time on Unit Disk Graphs*†

Fedor V. Fomin[1], Daniel Lokshtanov[2], Fahad Panolan[3], Saket Saurabh[4], and Meirav Zehavi[5]

1 **Department of Informatics, University of Bergen, Bergen, Norway**
fomin@ii.uib.no
2 **Department of Informatics, University of Bergen, Bergen, Norway**
daniello@ii.uib.no
3 **Department of Informatics, University of Bergen, Bergen, Norway**
fahad.panolan@ii.uib.no
4 **Department of Informatics, University of Bergen, Bergen, Norway; and The Institute of Mathematical Sciences, HBNI, Chennai, India**
saket@imsc.res.in
5 **Department of Informatics, University of Bergen, Bergen, Norway**
meirav.zehavi@ii.uib.no

Abstract

We give algorithms with running time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ for the following problems. Given an n-vertex unit disk graph G and an integer k, decide whether G contains

- a path on exactly/at least k vertices,
- a cycle on exactly k vertices,
- a cycle on at least k vertices,
- a feedback vertex set of size at most k, and
- a set of k pairwise vertex-disjoint cycles.

For the first three problems, no subexponential time parameterized algorithms were previously known. For the remaining two problems, our algorithms significantly outperform the previously best known parameterized algorithms that run in time $2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$. Our algorithms are based on a new kind of tree decompositions of unit disk graphs where the separators can have size up to $k^{\mathcal{O}(1)}$ and there exists a solution that crosses every separator at most $\mathcal{O}(\sqrt{k})$ times. The running times of our algorithms are optimal up to the $\log k$ factor in the exponent, assuming the Exponential Time Hypothesis.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations, G.2.2 Graph Algorithms

Keywords and phrases Longest Cycle, Cycle Packing, Feedback Vertex Set, Unit Disk Graph, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.65

* The full version of the paper can be found at arXiv [21], https://arxiv.org/abs/1704.07279.
† Supported by Pareto-Optimal Parameterized Algorithms, ERC Starting Grant 715744, Parameterized Approximation, ERC Starting Grant 306992, and Rigorous Theory of Preprocessing, ERC Advanced Investigator Grant 267959.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 65; pp. 65:1–65:15


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

1 Introduction

Unit disk graphs are the intersection graphs of unit circles in the plane. That is, given n-unit circles in the plane, we have a graph G where each vertex corresponds to a circle such that there is an edge between two vertices when the corresponding circles intersect. Unit disk graphs form one of the most well studied graph classes in computational geometry because of their use in modelling optimal facility location [40] and broadcast networks such as wireless, ad-hoc and sensor networks [25, 33, 42]. These applications have led to an extensive study of NP-complete problems on unit disk graphs in the realms of computational complexity and approximation algorithms. We refer the reader to [10, 18, 29] and the citations therein for these studies. However, these problems remain hitherto unexplored in the light of parameterized complexity with exceptions that are few and far between [1, 8, 23, 32, 38].

In this paper we consider the following basic problems about finding, hitting and packing cycles on unit disk graphs from the viewpoint of parameterized algorithms. For a given graph G and integer k,

- Exact k-Cycle asks whether G contains a cycle on exactly k vertices,
- Longest Cycle asks whether G contains a cycle on at least k vertices,
- Feedback Vertex Set asks whether G contains a vertex set S of size k such that the graph $G \setminus S$ is acyclic, and
- Cycle Packing asks whether G contains a set of k pairwise vertex-disjoint cycles.

Along the way, we also study Longest Path (decide whether G contains a path on exactly/at least k vertices) and Subgraph Isomorphism (SI). In SI, given *connected* graphs G and H on n and k vertices, respectively, the goal is to decide whether there exists a subgraph in G that is isomorphic to H. We also assume that a unit disk graph is given by a set of n points in $\mathbb{R}^2$ and there is a graph where vertices correspond to the points and there is an edge between two vertices if and only if the distance between the two points is at most 2.

In parameterized complexity each of these problems serves as a testbed for development of fundamental algorithmic techniques such as color-coding [2], the polynomial method [34, 35, 41, 4], matroid based techniques [20] for Longest Path and Longest Cycle, and kernelization techniques for Feedback Vertex Set [39]. We refer to [12] for an extensive overview of the literature on parameterized algorithms for these problems. For example, the fastest known algorithms solving Longest Path are the $1.66^k \cdot n^{\mathcal{O}(1)}$ time randomized algorithm of Björklund et al. [4], and the $2.597^k \cdot n^{\mathcal{O}(1)}$ time deterministic algorithm of Zehavi [43]. Moreover, unless the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi and Zane [30] fails, none of the problems above can be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ [30].

While all these problems remain NP-complete on planar graphs, substantially faster – *subexponential* – parameterized algorithms are known on planar graphs. In particular, by combining the bidimensionality theory of Demaine et al. [13] with efficient algorithms on graphs of bounded treewidth [17], Longest Path, Longest Cycle, Feedback Vertex Set and Cycle Packing are solvable in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ on planar graphs. The parameterized subexponential "tractability" of such problems can be extended to graphs excluding some fixed graph as a minor [15]. The bidimensionality arguments cannot be applied to Exact k-Cycle and this was one of the motivations for developing the new pattern-covering technique, which is used to give a randomized algorithm for Exact k-Cycle running in time $2^{\mathcal{O}(\sqrt{k}\log^2 k)}n^{\mathcal{O}(1)}$ on planar and apex-minor-free graphs [19]. The bidimensionality theory was also used to design (efficient) polynomial time approximation scheme ((E)PTAS) [14, 22] and polynomial kernelization [24] on planar graphs.

It would be interesting to find generic properties of problems, similar to the theory of bidimensionality for planar-graph problems, that could guarantee the existence of subexponential

parameterized algorithms or (E)PTAS on geometric classes of graphs, such as unit disk graphs. The theory of (E)PTAS on geometric classes of graphs is extremely well developed and several methods have been devised for this purpose. This includes methods such as shifting techniques, geometric sampling and bidimensionality theory [29, 27, 26, 28, 11, 37, 23]. However, we are still very far from a satisfactory understanding of the "subexponential" phenomena for problems on geometric graphs. We know that some problems such as Independent Set and Dominating Set, which are solvable in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ on planar graphs, are W[1]-hard on unit disk graphs and thus the existence of an algorithm of running time $f(k) \cdot n^{\mathcal{O}(1)}$ is highly unlikely for any function f [36]. The existence of a vertex-linear kernel for some problems on unit disk graphs such as Vertex Cover [9] or Connected Vertex Cover [32] combined with an appropriate separation theorem [1, 8, 38] yields a parameterized subexponential algorithm. A subset of the authors of this paper used a different approach based on bidimensionality theory to obtain subexponential algorithms of running time $2^{\mathcal{O}(k^{0.75}\log k)} \cdot n^{\mathcal{O}(1)}$ on unit disk graphs for Feedback Vertex Set and Cycle Packing in [23]. No parameterized subexponential algorithms on unit disk graphs for Longest Path, Longest Cycle, and Exact k-Cycle were known prior to our work.

Our Results. We design subexponential parameterized algorithms, with running time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$, for Exact k-Cycle, Longest Cycle, Longest Path, Feedback Vertex Set and Cycle Packing on unit disk graphs and unit square graphs. It is also possible to show by known NP-hardness reductions for problems on unit disk graphs [10] that an algorithm of running time $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ for any of our problems on unit disk graphs would imply that ETH fails. Hence our algorithms are asymptotically almost tight. Along the way we also design Turing kernels (in fact, many to one) for Exact k-Cycle, Longest Cycle, Longest Path and SI. That is, we give a polynomial time algorithm that given an instance of Exact k-Cycle or Longest Cycle or Longest Path or SI, produces polynomially many reduced instances of size polynomial in k such that the input instance is a Yes-instance if and only if one of the reduced instances is. As a byproduct of this we obtain a $2^{\mathcal{O}(k\log k)} \cdot n^{\mathcal{O}(1)}$ time algorithm for SI when G is a unit disk graph and H is an arbitrary connected graph. It is noteworthy to remark that a simple disjoint union trick implies that Exact k-Cycle, Longest Cycle, Longest Path, and SI do not admit a polynomial kernel on unit disk graphs [6]. Finally, we note that we do not use Turing kernels to design our subexponential time algorithms except for Exact k-Cycle. The subexponential time parameterized algorithm for Exact k-Cycle also uses a "double layering" of Baker's technique [3].

All our subexponential time algorithms have the following theme in common. If an input n-vertex unit disk graph G contains a clique of size $\mathsf{poly}(k)$ (such a clique can be found in polynomial time), then we have a trivial Yes-instance or No-instance, depending on the problem. Otherwise, we show that the unit disk graph G in a Yes-instance of the problem admits, sometimes after a polynomial time preprocessing, a specific type of (ω, Δ, τ)-decomposition, where the meaning of ω, Δ and τ is as follows. The vertex set of G is partitioned into cliques $C_1, \ldots, C_d$, each of size at most $\omega = k^{\mathcal{O}(1)}$. We also require that after contracting each of the cliques C_i to a single vertex, the maximum vertex degree Δ of the obtained graph $\tilde{G}$ is $\mathcal{O}(1)$, while the treewidth τ of $\tilde{G}$ is $\mathcal{O}(\sqrt{k})$. Moreover, the corresponding tree decomposition of $\tilde{G}$ can be constructed efficiently. We use the tree decomposition of $\tilde{G}$ to construct a tree decomposition of G by "uncontracting" each of the contracted cliques C_i. While the width of the obtained tree decomposition of G can be of order $\omega \cdot \tau = k^{\mathcal{O}(1)}$, we show that each of our parameterized problems can be solved in time $f(\Delta) \cdot \omega^{f(\Delta)\cdot\tau}$. Here we use dynamic programming over the constructed tree decomposition of G, however there

is a twist from the usual way of designing such algorithms. This part of the algorithm is problem-specific – in order to obtain the claimed running time, we have to establish a very specific property for each of the problems. Roughly speaking, the desired property of a problem is that it always admits an optimal solution such that for every pair of adjacent bags X, Y of the tree decomposition of G, the number of edges of this solution "crossing" a cut between X and Y is $\mathcal{O}(\sqrt{k})$. We remark that the above decomposition is *only* given in the introduction to present our ideas for all the algorithms in a unified way.

In this paper we restrict our attention to solving Exact k-Cycle and Feedback Vertex Set on unit disk graphs. We refer to the full version of the paper for all the proofs and the results.

2 Preliminaries

For a positive integer t, we use $[t]$ as a shorthand for $\{1, 2, \ldots, t\}$. Given a function $f : A \to B$ and a subset $A' \subseteq A$, let $f|_{A'}$ denote the restriction of the function f to the domain A'. For a function $f : A \to B$ and $B' \subseteq B$, $f^{-1}(B')$ denote the set $\{a \in A \ : \ f(a) \in B'\}$. For $t, t' \in \mathbb{N}$, a set $[t] \times [t']$, $i \in [t]$ and $j \in [t']$ we use $(*, j)$ and $(i, *)$ to denote the sets $\{(i', j) \ : \ i' \in [t]\}$ and $\{(i, j') \ : \ j' \in [t']\}$, respectively. For a set U, we use 2^U to denote the power set of U.

We use standard notation and terminology from the book of Diestel [16] for graph-related terms which are not explicitly defined here. Given a graph G, $V(G)$ and $E(G)$ denote its vertex-set and edge-set, respectively. When the graph G is clear from context, we denote $n = |V(G)|$ and $m = |E(G)|$. Given $U \subseteq V(G)$, we let $G[U]$ denote the subgraph of G induced by U, and we let $G \setminus U$ denote the graph $G[V(G) \setminus U]$. For an edge subset E, we use $V(E)$ to denote the set of endpoints of edges in E and $G[E]$ to denote the graph $(V(E), E)$. For $X, Y \subseteq V(G)$, we use $E(X)$ and $E(X, Y)$ to denote the edge sets $\{\{u, v\} \in E(G) \ : \ u, v \in X\}$ and $\{\{u, v\} \in E(G) \ : \ u \in X, v \in Y\}$, respectively. Moreover, we let $N(U)$ denote the open neighborhood of G. In case $U = \{v\}$, we denote $N(v) = N(U)$. Given an edge $e = \{u, v\} \in E(G)$, we use G/e to denote the graph obtained from G by contracting the edge e. In other words, G/e denotes the graph on the vertex-set $(V(G) \setminus \{u, v\}) \cup \{x_{\{u,v\}}\}$, where $x_{\{u,v\}}$ is a new vertex, and the edge-set $E(G) = E(G[V(G) \setminus \{u, v\}]) \cup \{\{x_{\{u,v\}}, w\} \mid w \in N(\{u, v\})\}$. A graph H is called a *minor* of G, if H can be obtained from G by a sequence of edge deletion, edge contraction and vertex deletion. Given $k \in \mathbb{N}$, we let K_k denote the complete graph on k vertices. For a set X, we use $K[X]$ to denote the complete graph on X. Given $a, b \in \mathbb{N}$, an $a \times b$ grid is a graph on $a \cdot b$ vertices, $v_{i,j}$ for $(i, j) \in [a] \times [b]$, such that for all $i \in [a - 1]$ and $j \in [b]$, it holds that $v_{i,j}$ and $v_{i+1,j}$ are neighbors, and for all $i \in [a]$ and $j \in [b - 1]$, it holds that $v_{i,j}$ and $v_{i,j+1}$ are neighbors. For ease of presentation, for any function $f : D \to [a] \times [b]$, $i \in [a]$ and $j \in [b]$, we use $f^{-1}(i, j)$, $f^{-1}(*, j)$, and $f^{-1}(i, *)$ to denote the sets $f^{-1}((i, j))$, $f^{-1}((*, j))$, and $f^{-1}((i, *))$, respectively.

We also need the standard notions of pathwidth, treewidth and nice tree decomposition. These definitions are given in the appendix for easy perusal (also in the appended version). However, we use slightly different but equivalent definition of path decomposition.

▶ **Definition 1.** A *path decomposition* of a graph G is a sequence $\mathcal{P} = (X_1, X_2, \ldots, X_\ell)$, where each $X_i \subseteq V(G)$ is called a *bag*, that satisfies the following conditions.

- $\bigcup_{i \in [\ell]} X_i = V(G)$.
- For every edge $\{u, v\} \in E(G)$ there exists $i \in [\ell]$ such that $\{u, v\} \subseteq X_i$.
- For every vertex $v \in V(G)$, if $v \in X_i \cap X_j$ for some $i \leq j$, then $v \in X_r$ for all $r \in \{i, \ldots, j\}$.

The *width* of $\mathcal{P}$ is $\max_{i \in [\ell]} |X_i| - 1$.

The *pathwidth* of G, $\mathsf{pw}(G)$, is the minimum width of a path decomposition of G.

▶ **Proposition 2** ([5, 7]). *Given a graph G and an integer k, in time $2^{\mathcal{O}(k)} \cdot n$, we can either decide that $\mathsf{tw}(G) > k$ or output a tree decomposition of G of width $5k$. Furthermore, given a graph G and a tree decomposition $\mathcal{T}$ of G, a nice tree decomposition $\mathcal{T}'$ of the same width as $\mathcal{T}$ can be computed in linear time.*

Given a set of geometric objects, O, we say that a graph G *represents* O if each vertex in $V(G)$ represents a distinct geometric object in O, and every geometric object in O is represented by a distinct vertex in $V(G)$. In this case, we abuse notation and write $V(G) = O$. The *intersection graph of* O is a graph G that represent O and satisfies $E(G) = \{\{u, v\} : u, v \in O, u \cap v \neq \emptyset\}$. Let $P = \{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)\}$ be a set of points in the Euclidean plane. In the *unit disk graph model*, for every $i \in [n]$, we let d_i denote the disk of radius 1 whose centre is p_i. Accordingly, we denote $D = \{d_1, d_2, \ldots, d_n\}$. Then, the *unit disk graph of* D is the intersection graph of D. Alternatively, the unit disk graph of D is the geometric graph of G such that $E(G) = \{\{p_i = (x_i, y_i), p_j = (x_j, y_j)\} \mid p_i, p_j \in D, i \neq j, \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq 2\}$.

3 Clique-Grid Graphs

In this section, we introduce a family of "grid-like" graphs, called clique-grid graphs, that is tailored to fit our techniques. Given a unit disk graph G, we extract the properties of G that we would like to exploit, and show that they can be captured by an appropriate clique-grid graph. Let us begin by giving the definition of a clique-grid graph. Roughly speaking, a graph G is a clique-grid graph if each of its vertices can be embedded into a single cell of a grid (where multiple vertices can be embedded into the same cell), ensuring that the subgraph induced by each cell is a clique, and that each cell can interact (via edges incident to its vertices) only with cells at "distance" at most 2. Formally,

▶ **Definition 3** (clique-grid graph). A graph G is a *clique-grid graph* if there exists a function $f : V(G) \to [t] \times [t']$, for some $t, t' \in \mathbb{N}$, such that the following conditions are satisfied.

1. For all $(i, j) \in [t] \times [t']$, it holds that $f^{-1}(i, j)$ is a clique.
2. For all $\{u, v\} \in E(G)$, it holds that if $f(u) = (i, j)$ and $f(v) = (i', j')$ then $|i - i'| \leq 2$ and $|j - j'| \leq 2$.

Such a function f is a *representation* of G.

We note that a notion similar to clique-grid graph was also used by Ito and Kadoshita [31]. For the sake of clarity, we say that a pair $(i, j) \in [t] \times [t']$ is a *cell*. The following lemma states that a unit disk graph is a clique-grid graph, and its proof is deferred to [21].

▶ **Lemma 4.** *Let D be a set of points in the Euclidean plane, and let G be the unit disk graph of D. Then, a representation f of G can be computed in polynomial time.*

Due to Lemma 4, throughout this paper, we assume that along with a input unit disk graph G, we are given a representation f of G. We conclude this section by introducing the definition of an ℓ-NCTD which is useful for doing our dynamic programming algorithms.

▶ **Definition 5.** A tree decomposition $\mathcal{T} = (T, \beta)$ of a clique-grid graph G with representation f is a *nice ℓ-clique tree decomposition*, or simply an *ℓ-NCTD*, if for the root r of T, it holds that $\beta(r) = \emptyset$, and for each node $v \in V(T)$, it holds that

- There exist at most ℓ cells, $(i_1, j_1), \ldots, (i_\ell, j_\ell)$, such that $\beta(v) = \bigcup_{t=1}^{\ell} f^{-1}(i_t, j_t)$, and
- The node v is of one of the following types.

- **Leaf**: v is a leaf in T and $\beta(v) = \emptyset$.
- **Forget**: v has exactly one child u, and there exists a cell $(i,j) \in [t] \times [t']$ such that $f^{-1}(i,j) \subseteq \beta(u)$ and $\beta(v) = \beta(u) \setminus f^{-1}(i,j)$.
- **Introduce**: v has exactly one child u, and there exists a cell $(i,j) \in [t] \times [t']$ such that $f^{-1}(i,j) \subseteq \beta(v)$ and $\beta(v) \setminus f^{-1}(i,j) = \beta(u) \setminus f^{-1}(i,j)$.
- **Join**: v has exactly two children, u and w, and $\beta(v) = \beta(u) = \beta(w)$.

A *nice ℓ-clique path decomposition*, or simply an *ℓ-NCPD*, is an ℓ-NCTD where T is a path. In this context, for convenience, we use the notation referring to a sequence presented in Section 2.

4 Turing Kernels

In this section we give an overview of a Turing kernel (actually a compression) for SI. The reason for stating our result in this way is, that this is how we use it in the next section to design a subexponential algorithm for Exact k-Cycle. We refer to the appended version for a Turing kernel for SI and Longest Cycle. More precisely, we show the following.

▶ **Theorem 6.** *Let (G, f, H, k) be an instance of* SI *on unit disk graphs, then in polynomial time, one can output a set $\mathcal{I}$ of instances of* SI *on clique-grid graphs such that (G, f, H, k) is a* Yes*-instance if and only if at least one instance in $\mathcal{I}$ is a* Yes*-instance, and for all $(\widehat{G}, \widehat{f} : V(\widehat{G}) \to [\widehat{t}] \times [\widehat{t'}], \widehat{H}, \widehat{k}) \in \mathcal{I}$, $\widehat{G}$ is either an induced subgraph of G or $K_{\widehat{k}}$, $\widehat{t}, \widehat{t'} \leq 2\widehat{k}$, $|\widehat{f}^{-1}(i,j)| < \widehat{k}$ for any $(i,j) \in [\widehat{t}] \times [\widehat{t'}]$, $\widehat{H} = H$, $\widehat{k} = k$, $|V(\widehat{G})| = \mathcal{O}(k^3)$, and $|E(\widehat{G})| = \mathcal{O}(k^4)$.*

Proof Sketch. First, suppose that there exists a cell $(i,j) \in [t] \times [t']$ such that $|f^{-1}(i,j)| \geq k$, then by Definition 3, $G[f^{-1}(i,j)]$ is a clique on at least k vertices. In particular, the pattern H is a subgraph of $G[f^{-1}(i,j)]$, and therefore it is also a subgraph of G. Thus, in this case, we conclude the proof by setting $\mathcal{I}$ to be the set that contains only one instance, $(K_k, \widehat{f} : V(K_k) \to [1] \times [1], H, k)$. From now on, suppose that for all cells $(i,j) \in [t] \times [t']$, it holds that $|f^{-1}(i,j)| < k$.

Now, our kernelization algorithm works as follows. For every $(p,q) \in [t] \times [t']$, it computes

$$G_{p,q} = G[\bigcup_{\substack{p \leq i < \min\{p+2k, t+1\} \\ q \leq j < \min\{q+2k, t'+1\}}} f^{-1}(i,j)].$$

Accordingly, it computes $f_{p,q} : V(G_{p,q}) \to [\min\{2k,t\}] \times [\min\{2k,t'\}]$ as follows. For every $v \in V(G_{p,q})$, compute $f_{p,q}(v) = (i-p+1, j-q+1)$ where $(i,j) = f(v)$. Note that for all $(i,j) \in [\min\{2k,t\}] \times [\min\{2k,t'\}]$, it holds that $f_{p,q}^{-1}(i,j) = f^{-1}(i+p-1, j+q-1)$. Thus, since f is a representation of G, it holds that $f_{p,q}$ is a representation of $G_{p,q}$. Finally, our kernelization algorithm outputs $\mathcal{I} = \{I_{p,q} = (G_{p,q}, f_{p,q}, H, k) : (p,q) \in [t] \times [t']\}$.

To conclude the proof, it remains to show that (G, f, H, k) is a Yes-instance if and only if at least one instance in $\mathcal{I}$ is a Yes-instance. Since for all $(G_{p,q}, f_{p,q}, H, k) \in \mathcal{I}$, it holds that $G_{p,q}$ is an induced subgraph of G, we have that if (G, f, H, k) is a No-instance, then every instance in $\mathcal{I}$ is No-instance as well. Next, suppose that (G, f, H, k) is a Yes-instance. Then, let H' be a subgraph of G that is isomorhpic to H. In order to complete the proof, we introduce a notion of *ℓ-stretched*. We say that H' is *ℓ-stretched* if there exist cells $(i,j), (i',j') \in [t] \times [t']$ such that the following conditions are satisfied: (i) $|i-i'| \geq 2\ell$ or $|j-j'| \geq 2\ell$ (or both); and (b) $V(H') \cap f^{-1}(i,j) \neq \emptyset$ and $V(H') \cap f^{-1}(i',j') \neq \emptyset$. One can show (see the appended version) that for any subgraph H' of G that is isomorphic to H, it holds that H' is not $2k$-stretched. Using this *claim* we can conclude. Denote $i_{\min} = \min\{i \in [t] : (\bigcup_{j \in [t']} f^{-1}(i,j)) \cap V(H') \neq \emptyset\}$,

$i_{\max} = \max\{i \in [t] : (\bigcup_{j\in[t']} f^{-1}(i,j)) \cap V(H') \neq \emptyset\}$, $j_{\min} = \min\{j \in [t'] : (\bigcup_{i\in[t]} f^{-1}(i,j)) \cap V(H') \neq \emptyset\}$ and $j_{\max} = \max\{j \in [t'] : (\bigcup_{i\in[t]} f^{-1}(i,j)) \cap V(H') \neq \emptyset\}$. From the above claim, it holds that both $i_{\max} - i_{\min} < 2k$ and $j_{\max} - j_{\min} < 2k$. Therefore, H' is a subgraph of $G_{i_{\min},j_{\min}}$, which implies that $I_{p,q}$ is a YES-instance. ◀

5 Exact k-Cycle

In this section we prove the following theorem.

▶ **Theorem 7.** EXACT k-CYCLE *on unit disk graphs can be solved in* $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ *time.*

By Theorem 6, we have seen that SI admits a polynomial sized Turing kernel. Hence to give an algorithm of running time $2^{\mathcal{O}(\sqrt{k}\log k)}|V(G)|^{\mathcal{O}(1)}$, we can restrict to instances returned by Theorem 6. More precisely, because of Theorem 6, we can assume that the input to our algorithm is $(G, f : V(G) \to [t] \times [t'], k)$ where G is a clique-grid graph with a representation f, $|f^{-1}(i,j)| < k$ for all $(i,j) \in [t] \times [t']$, and $t, t' \leq 2k$. Without loss of generality we can assume that f is a function from $V(G)$ to $[2k] \times [2k]$, because $[t] \times [t'] \subseteq [2k] \times [2k]$.

Given an instance $(G, f : V(G) \to [2k] \times [2k], k)$, the algorithm applies a method inspired by Baker's technique [3] and obtains a family, $\mathcal{F}$, of $2^{\mathcal{O}(\sqrt{k}\log k)}$ instances of EXACT k-CYCLE. The family $\mathcal{F}$ has following properties.

1. In each of these instances the input graph is an induced subgraph of G and has size $k^{\mathcal{O}(1)}$.
2. The input $(G, f : V(G) \to [2k] \times [2k], k)$ is a YES-instance if and only if there exists an instance $(H, f^* : V(H) \to [2k] \times [2k], k) \in \mathcal{F}$ which is a YES-instance.
3. More over, for any instance $(H, f^* : V(H) \to [2k] \times [2k], k) \in \mathcal{F}$, H has a nice $7\sqrt{k}$-clique path decomposition ($7\sqrt{k}$-NCPD).

We will call the family $\mathcal{F}$ satisfying the above properties as *good family*. Let $(H, f^* : V(H) \to [2k] \times [2k], k)$ be an instance of $\mathcal{F}$. Let $\mathcal{P} = (X_1, \ldots, X_q)$ be a $7\sqrt{k}$-NCPD of H. We first prove that if there is a cycle of length k in H, then there is a cycle C of length k in H such that for any two distinct cells (i,j) and (i',j') of f, the number of edges with one end point in (i,j) and other in (i',j') is at most 4. Let C be such a cycle in H. Then using the property of C we get the following important property.

> For any $i \in [q]$, the number of edges of $V(C)$ with one end point in X_i and other in $\bigcup_{i<j\leq q} X_j$ is in $\mathcal{O}(\sqrt{k})$.

The above mentioned property allows us to design a dynamic programming (DP) algorithm over $7\sqrt{k}$-NCPD, $\mathcal{P}$, for EXACT k-CYCLE in time $2^{\mathcal{O}(\sqrt{k}\log k)}$. Now we are ready to give formal details about the algorithm. As explained before, we assume that the number of vertices in the input graph is bounded by $k^{\mathcal{O}(1)}$.

▶ **Lemma 8.** *Let* $(G, f : V(G) \to [2k] \times [2k], k)$ *be an instance of* EXACT k-CYCLE*, where* G *is a clique-grid graph with representation* f*,* $|f^{-1}(i,j)| < k$ *for all* $(i,j) \in [2k] \times [2k]$ *and* $|V(G)| = k^{\mathcal{O}(1)}$*. Given* $(G, f : V(G) \to [2k] \times [2k], k)$*, there is an algorithm running in time* $2^{\mathcal{O}(\sqrt{k}\log k)}$ *that outputs a good family* $\mathcal{F}$*.*

Proof. Let C be a k length cycle in G. First we define a column of the $2k \times 2k$ grid. For any $j \in [2k]$ the set of cells $\{(i,j) \ : \ i \in [2k]\}$ is called a column. There are $2k$ columns for the $2k \times 2k$ grid. We partition $2k$ columns of the $2k \times 2k$ grid with k blocks of two consecutive columns and label them from the set of labels $[\sqrt{k}]$. That is, both columns $2i-1$ and $2i$,

where $i \in [k]$, are labelled with $i \mod \sqrt{k}$. Then by pigeon hole principle there is a label $\ell \in \{1, 2, \ldots, \sqrt{k}\}$ such that the number of vertices from $V(C)$ which are in columns labelled ℓ is at most $\sqrt{k}$. As $|V(G)| \leq k^{\mathcal{O}(1)}$, the number of vertices of G in columns labelled ℓ is at most $k^{\mathcal{O}(1)}$. We guess the vertices of $V(C)$ which are in the columns labelled ℓ. The number of potential guesses is bounded by $k^{\mathcal{O}(\sqrt{k})}$. Let Y be the set of guessed vertices of $V(C)$ which are in the columns labelled by ℓ. Notice that $|Y| \leq \sqrt{k}$. Then we delete all the vertices in columns labelled ℓ, except the vertices of Y. Let S be the set of deleted vertices. By the property 2 of clique-grid graph, $G \setminus (S \cup Y)$ is a disjoint union of clique-grid graphs each of which is represented by a function with at most $2\sqrt{k}$ columns. That is, let $G_1 = G[\bigcup_{j=1}^{2(\ell-1)} f^{-1}(*, j)]$, and $G_{i+1} = G[\bigcup_{j=i\cdot 2\ell+1}^{\min\{i\cdot 2\ell+2\sqrt{k}, 2k\}} f^{-1}(*, j)]$ for all $i \in \{1, \ldots, \lceil\sqrt{k}\rceil\}$. Notice that G_i is clique-grid graph with representation $f_i : V(G_i) \to [2k] \times [2\sqrt{k}]$ defined as, $f_i(u) = (r, j)$, when $f(u) = (r, (i-1)2\ell+j)$. By the property 2 of clique-grid graph, $G \setminus (S \cup Y) = G_1 \uplus \ldots \uplus G_{\lceil\sqrt{k}\rceil+1}$.

▶ **Claim 1.** *$G \setminus S$ has a nice $7\sqrt{k}$-clique path decomposition ($7\sqrt{k}$-NCPD).*

Proof. First, for each $i \in \{1, \ldots, \lceil\sqrt{k}\rceil + 1\}$, we will define a path decomposition of G_i (in the next paragraph) such that each bag is a union of at most $6\sqrt{k}$ many cells of f_i. As $G \setminus (S \cup Y) = G_1 \uplus \ldots \uplus G_{\lceil\sqrt{k}\rceil-1}$, and $|Y| \leq \sqrt{k}$, by adding Y to each bag of all path decompositions we can get a required nice $7\sqrt{k}$-clique path decomposition for $G \setminus S$.

Now, for each G_i, we define a path decomposition $\mathcal{P}_i = (X_{i,1}, X_{i,2}, \ldots X_{i,2k-2})$ where $X_{i,j} = f_i^{-1}(j, *) \cup f_i^{-1}(j+1, *) \cup f_i^{-1}(j+2, *)$. We claim that $\mathcal{P}_i$ is a path decomposition of G_i. Notice that $\bigcup_{j=1}^{k-1} X_{i,j} = f_i^{-1}(*, *) = V(G_i)$. By property 2 of clique-grid graph, we have that for each edge $\{u, v\} \in E(G)$, there exists $j \in [2k-2]$ such that $\{u, v\} \in X_{i,j}$. For each $u \in V(G)$, u is contained in at most three bags and these bags are consecutive in the sequence $(X_{i,1}, X_{i,2}, \ldots X_{i,2k-2})$. Hence $\mathcal{P}_i$ is a path decomposition of G_i. Since $X_{i,j} = f_i^{-1}(j, *) \cup f_i^{-1}(j+1, *) \cup f_i^{-1}(j+2, *)$, number of columns in f_i is at most $2\sqrt{k}$ and each cell of f_i is a cell of f, each $X_{i,j}$ is a union of $6\sqrt{k}$ many cells of f. Since $G \setminus (S \cup Y) = G_1 \uplus \ldots \uplus G_{\lceil\sqrt{k}\rceil+1}$, the sequence $\mathcal{P}' = (X_{1,1}, \ldots X_{1,2k-2}, X_{2,1}, \ldots X_{2,2k-2}, \ldots, X_{\lceil\sqrt{k}\rceil-1,1}, \ldots X_{\lceil\sqrt{k}\rceil-1,2k-2})$ is a path decomposition of $G \setminus (S \cup Y)$. More over, the vertices of each bag is a union of vertices from at most $6\sqrt{k}$ cells of f. Also, since $|Y| \leq \sqrt{k}$, the sequence $\mathcal{P} = (X_{1,1} \cup Y, \ldots X_{1,k-2} \cup Y, \ldots X_{\lceil\sqrt{k}\rceil-1,k-2} \cup Y)$ obtained by adding Y to each bag of $\mathcal{P}'$ we get a path decomposition of $G \setminus S$. More over, the vertices of each bag in $\mathcal{P}$ is a union of vertices from at most $7\sqrt{k}$ cells of f. We can turn the path decomposition $\mathcal{P}$ to a $7\sqrt{k}$-NCPD by an algorithm similar to the one mentioned in Proposition 2. ◀

The algorithm constructs a family $\mathcal{F}$ as follows. For each $\ell \in \{1, \ldots, \lceil\sqrt{k}\rceil\}$ and for two subsets of vertices S and Y such that $S \cup Y$ is a set of vertices in the columns labelled ℓ and $|Y| \leq \sqrt{k}$, our algorithm includes an instance $(G \setminus S, f|_{V(G)\setminus S}, k)$ in $\mathcal{F}$. The number of choices of S and Y is bounded by $2^{\mathcal{O}(\sqrt{k}\log k)}$ and thus the size of $\mathcal{F}$ is bounded by $2^{\mathcal{O}(\sqrt{k}\log k)}$. We claim that $\mathcal{F}$ is indeed a good family. Let C be a cycle of length k in G. Then, there is an $\ell \in \{1, \ldots, \lceil\sqrt{k}\rceil\}$ such that at most $\sqrt{k}$ vertices from $V(C)$ are in the columns labelled by ℓ. Let S' be the set of vertices in the columns labelled by ℓ. Let $Y = S' \cap V(C)$ and $S = S' \setminus Y$. The instance $(G \setminus S, f|_{V(G)\setminus S}, k)$ in $\mathcal{F}$ is a YES instance. This concludes the proof. ◀

Now we can assume that we are solving EXACT k-CYCLE on (H, f, k), where $(H, f, k) \in \mathcal{F}$ (here we rename the function $f|_{V(H)}$ with f for ease of presentation). Now we prove that if there is a cycle of length k in H, then there is a cycle C of length k in H such that for any two cells (i, j) and (i', j') of f, the number of edges of $E(C)$ with one end point in (i, j) and other (i', j') is at most 5.

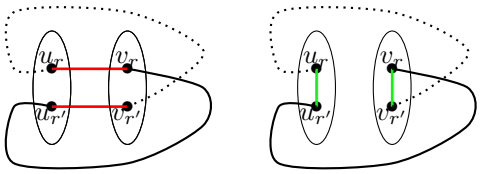


Figure 1 Illustration of Lemma 9. Figure on the left is the cycle $C = [u_r v_r]Q_1[u_{r'} v_{r'}]Q_2$ and the one on the right is the cycle $C' = [u_r u_{r'}]\overleftarrow{Q}_1[v_r v_{r'}]Q_2$.

▶ **Lemma 9.** *Let $(H, f : V(H) \to [2k] \times [2k], k)$ be a* Yes *instance of* Exact k-Cycle. *Then there is a cycle C of length k in H such that for any two distinct cells (i, j) and (i', j') of f, the number of edges of $E(C)$ with one end point in (i, j) and other (i', j') is at most* 5.

Proof. Let C be a k length cycle such that the number edges of $E(C)$ whose end points are in different cells is minimized. We claim that for any two disjoint cells (i, j) and (i', j'), the number of edges of $E(C)$ with one end point in (i, j) and other (i', j') is at most 5. Suppose not. Then there exist (i, j) and (i', j') such that the number of edges of $E(C)$ with one end point in (i, j) and other in (i', j') is at least 6. Let $C = P_1[u_1 v_1]P_2[u_2 v_2]P_3[u_3 v_3]P_4[u_4 v_4]P_5[u_5 v_5]P_6[u_6 v_6]$ where for each $\{u_r, v_r\}, r \in [6]$, one end point is in the cell (i, j) and other in the cell $(i'j')$, and each subpath $P_\ell, \ell \in [6]$, can be empty too. Since C is a cycle, at least 3 edges from $\{\{u_r, v_r\} : i \in [6]\}$ form a matching. Let $\{u_{r_1}, v_{r_1}\}, \{u_{r_2}, v_{r_2}\}$ and $\{u_{r_3}, v_{r_3}\}$ be a matching of size 3, where $\{r_1, r_2, r_3\} \subseteq [6]$. Then, by pigeon hole principle there exist $r, r' \in \{r_1, r_2, r_3\}$ such that either $u_r, u_{r'} \in f^{-1}(i, j)$ or $u_r, u_{r'} \in f^{-1}(i', j')$. Without loss of generality assume that $u_r, u_{r'} \in f^{-1}(i, j)$ (otherwise we rename cell (i, j) with (i', j') and vice versa). That is, $C = [u_r v_r]Q_1[u_{r'} v_{r'}]Q_2$ such that $u_r, u_{r'} \in f^{-1}(i, j)$ and $v_r, v_{r'} \in f^{-1}(i', j')$. Then, since $f^{-1}(i, j)$ and $f^{-1}(i', j')$ are cliques, $C' = [u_r u_{r'}]\overleftarrow{Q}_1[v_r v_{r'}]Q_2$ is a k length cycle in G, such that the number edges of $E(C')$ whose end points are in different cells is less than that of $E(C)$, which is contradiction to our assumption. See Fig. 1 for an illustration of C and C'. This completes the proof. ◀

Next we design a DP algorithm that finds a cycle of length k, if it exists, satisfying properties of Lemma 9.

▶ **Lemma 10.** *Let $(H, f : V(H) \to [2k] \times [2k], k) \in \mathcal{F}$ be an instance of* Exact k-Cycle. *and $\mathcal{P}$ be a $7\sqrt{k}$-NCPD of H. Then, given $(H, f : V(H) \to [2k] \times [2k], k)$ and $\mathcal{P}$, there is an algorithm $\mathcal{A}$ which runs in time $2^{\mathcal{O}(\sqrt{k} \log k)}$, and outputs* Yes, *if there is a cycle C in H such that for any two distinct cells (i, j) and (i', j') of f, the number of edges with one end point in (i, j) and other (i', j') is at most* 5. *Otherwise algorithm $\mathcal{A}$ will output* No.

Proof Sketch. Algorithm $\mathcal{A}$ is a DP algorithm over the $7\sqrt{k}$-NCPD $\mathcal{P} = (X_1, \ldots X_q)$ of H. For any $\ell \in [q]$, we define H_ℓ be the induced subgraph $H[\bigcup_{i \leq \ell} X_i]$ of H. Define $\mathcal{C}$ to be the set of k-length cycles in H such that for any $C \in \mathcal{C}$ and two disjoint cells (i, j) and (i', j') of f, the number of edges of $E(C)$ with one end point in (i, j) and other (i', j') is at most 5. Let $C \in \mathcal{C}$. Since $\mathcal{P}$ is a $7\sqrt{k}$-NCPD and the fact that for any two distinct cells (i, j) and (i', j') of f, the number of edges of C with one end point in (i, j) and other (i', j') is at most 5, we have that for any bag X_ℓ of $\mathcal{P}$, the number of vertices of $V(C) \cap X_\ell$ which has a neighbour in $V(H) \setminus X_\ell$ is bounded by $\mathcal{O}(\sqrt{k})$. This allows us to keep only $2^{\mathcal{O}(\sqrt{k} \log k)}$ states in the DP algorithm. Fix any $\ell \in [q]$ and define C_L the set of paths of C (or the cycle C itself) when we restrict C to H_ℓ. That is $C_L = H_\ell[E(C)]$. Let $\widehat{C}_L = \{\{u, v\} \mid \text{there is a } u\text{-}v \text{ path in } C_L\}$. Notice that $\bigcup_{P \in \widehat{C}_L} P$ is the set of vertices of degree 0 or 1 in C_L and $\bigcup_{P \in \widehat{C}_L} P \subseteq X_\ell$. Since

X_ℓ is a union of vertices from at most $7\sqrt{k}$ many cells of f and for any two distinct cells (i,j) and (i',j') of f, the number of edges of $E(C)$ with one end point in (i,j) and other (i',j') is at most 5, and by property 2 of the clique-grid graph, we have that the cardinality of $\bigcup_{P \in \widehat{C}_L} P$ is at most $5 \cdot 24 \cdot 7\sqrt{k} = 840\sqrt{k}$. In our DP algorithm we will have state indexed by $(\ell, \widehat{C}_L, |E(C_L)|)$ which will be set to 1. Formally, for any $\ell \in [q]$, $k' \in [k]$ and a family $\mathcal{Z}$ of vertex disjoint sets of size at most 2 of X_ℓ with the property that the cardinality of $\bigcup_{Z \in \mathcal{Z}} Z$ is at most $840\sqrt{k}$, we will have a DP table entry $\mathcal{A}[\ell, \mathcal{Z}, k']$. For each $\ell \in [q]$, we maintain the following correctness invariant.

Correctness Invariants: (i) For every $C \in \mathcal{C}$, let $C_L = H_\ell[E(C)]$. For every $C \in \mathcal{C}$, let $\widehat{C}_L = \{\{u,v\} \mid$ there is a connected component P in C_L and P is a u-v path as well$\}$. Then $\mathcal{A}[\ell, \widehat{C}_L, |E(C_L)|] = 1$, (ii) for any family $\mathcal{Z}$ of vertex disjoint sets of size at most 2 of X_ℓ with $0 < |\bigcup_{Z \in \mathcal{Z}} Z| \leq 840\sqrt{k}$, $k' \in [k]$, and $\mathcal{A}[\ell, \mathcal{Z}, k'] = 1$, there is a set $\mathcal{Q}$ of $|\mathcal{Z}|$ vertex disjoint paths in H_ℓ where the end points of each path are specified by a set in $\mathcal{Z}$ and $|E(\mathcal{Q})| = k'$, and (iii) if $\mathcal{A}[\ell, \emptyset, k] = 1$, then there is a cycle of length k in H_ℓ.

The correctness of the our algorithm will follow from the correctness invariant. We fill the DP table entries similar to the way it is done for dynamic programming algorithms on graphs of bounded treewidth. That is, we fill the table entries by considering various cases for bags (introduce and forget) and using the previously computed DP table entries. A complete detailed proof is provided in the appended version. ◀

Theorem 7 follows from Theorem 6 and Lemmata 8, 9, and 10.

6 Feedback Vertex Set

In this section, we show that Feedback Vertex Set (FVS) admits a subexponential-time parameterized algorithm. More precisely, we prove the following.

▶ **Theorem 11.** FVS *on unit disk graphs can be solved in time* $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$.

For an algorithm for FVS and other problems such as Cycle Packing, we need more compact representations of clique-grid graphs. In the next section we introduce these notions.

6.1 The Cell Graph of a Clique-Grid Graph

We introduce two compact representations of clique-grid graphs. By examining these representations, we are able to infer information on the structure of clique-grid graphs that are also unit disk graphs.

▶ **Definition 12** (backbone)**.** Given a clique-grid graph G with representation $f : V(G) \to [t] \times [t']$, an induced subgraph H of G is a *backbone* for (G, f) if for any two distinct cells $(i,j), (i',j') \in [t] \times [t']$ for which there exist $u \in f^{-1}(i,j)$ and $v \in f^{-1}(i',j')$ such that $\{u,v\} \in E(G)$, there also exist $u' \in f^{-1}(i,j)$ and $v' \in f^{-1}(i',j')$ such that $\{u',v'\} \in E(H)$. If no induced subgraph of H is a backbone for (G,f), then H is a *minimal backbone* for (G,f).

First, we bound the maximum degree of a minimal backbone.

▶ **Lemma 13.** *Let G be a clique-grid graph with representation f, and let H be a minimal backbone for (G,f). Then, for all $(i,j) \in [t] \times [t']$, it holds that $|f^{-1}(i,j) \cap V(H)| \leq 24$. Furthermore, the maximum degree of H is at most* 599.

Proof. By Condition 2 in Definition 3, we have that for all cells $(i,j) \in [t] \times [t']$, it holds that $f^{-1}(i,j) \cap V(H) \leq |\{(i',j') \in [t] \times [t'] \setminus \{(i,j)\} \mid |i-i'| \leq 2, |j-j'| \leq 2\}| \leq 24$. Thus, for all $(i,j) \in [t] \times [t']$, the degree in H of a vertex in $f^{-1}(i,j) \cap V(H)$ is bounded by $|(\bigcup_{\{(i',j') \in [t] \times [t'] \mid |i-i'| \leq 2, |j-j'| \leq 2\}} f^{-1}(i,j) \cap V(H)) \setminus \{v\}| \leq |\{(i',j') \in [t] \times [t'] \mid |i-i'| \leq 2, |j-j'| \leq 2\}| \cdot 24 - 1 = 25 \cdot 24 - 1 = 599$. ◀

We compute a minimal backbone as follows. Initializes $H = G$; then, for every vertex $v \in V(G)$, it checks if the graph $H \setminus \{v\}$ has the same backbone as H, in which case it updates H to $H \setminus \{v\}$. Thus, a minimal backbone H for (G,f) can be computed in polynomial time. To analyze the treewidth of a backbone, we need the following.

▶ **Proposition 14** ([23]). *Let G be a unit disk graph with maximum degree Δ. Then G contains a $\frac{\mathsf{tw}}{100\Delta^3} \times \frac{\mathsf{tw}}{100\Delta^3}$ grid as a minor.*

▶ **Lemma 15.** *Given a clique-grid graph G that is a unit disk graph, a representation f of G and an integer $\ell \in \mathbb{N}$, in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$, one can either correctly conclude that G contains a $\frac{\ell}{100 \cdot 599^3} \times \frac{\ell}{100 \cdot 599^3}$ grid as a minor, or obtain a minimal backbone H for (G,f) with a nice tree decomposition $\mathcal{T}$ of width at most 5ℓ.*

We will use Lemma 15 with $\ell = \mathcal{O}(\sqrt{k})$. Next, we define a more compact representation of a clique-grid graph.

▶ **Definition 16** (cell graph). Given a clique-grid graph G with representation $f : V(G) \to [t] \times [t']$, the *cell graph* of G, denoted by $\mathsf{cell}(G)$, is the graph on the vertex-set $\{v_{i,j} : i \in [t], j \in [t'], f^{-1}(i,j) \neq \emptyset\}$ and edge-set $\{\{v_{i,j}, v_{i',j'}\} : (i,j) \neq (i',j'), \exists u \in f^{-1}(i,j) \exists v \in f^{-1}(i',j')$ such that $\{u,v\} \in E(G)\}$.

By Definitions 12 and 16, and the that for any graph G and a minor H of G, it holds that $\mathsf{tw}(H) \leq \mathsf{tw}(G)$, we conclude the following.

▶ **Observation 17.** *For a clique-grid graph G, a representation f of G and a backbone H for (G,f), it holds that $\mathsf{cell}(G)$ is a minor of H and $\mathsf{tw}(\mathsf{cell}(G)) \leq \mathsf{tw}(H)$.*

Note that a nice tree decomposition of $\mathsf{cell}(G)$ of width 5ℓ corresponds to a 5ℓ-NCTD of G. In other words, from Lemma 15 and Observation 17, we directly have the following..

▶ **Corollary 18.** *Given a clique-grid graph G that is a unit disk graph, a representation f of G and an integer $\ell \in \mathbb{N}$, in time $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$, one can either correctly conclude that G contains a $\frac{\ell}{100 \cdot 599^3} \times \frac{\ell}{100 \cdot 599^3}$ grid as a minor, or compute a 5ℓ-NCTD of G.*

6.2 Outline of an Algorithm for FVS

First, we observe that if we find a large grid, we can answer No (see also [15, 12]).

▶ **Observation 19.** *Let (G,k) be an instance of* FVS. *If G contains a $2\sqrt{k} \times 2\sqrt{k}$ grid as a minor, then (G,k) is a* No*-instance.*

This observation leads us to the following.

▶ **Lemma 20.** *Let (G,O,k) be an instance of* FVS *on unit disk graphs. Then, in time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot |V(G)|^{\mathcal{O}(1)}$, one can either solve (G,O,k) or obtain an equivalent instance (G,f,k) of* FVS *on clique-grid graphs together with an $\mathcal{O}(\sqrt{k})$-NCTD of G.*

Proof. First, by using Lemma 4, we obtain a representation f of G. Then, by using Corollary 18 with $\ell = 200 \cdot 599^3 \cdot \sqrt{k} = \mathcal{O}(\sqrt{k})$, we either correctly conclude that G contains a $2\sqrt{k} \times 2\sqrt{k}$ grid as a minor, or compute an $\mathcal{O}(\sqrt{k})$-NCTD of G. In both cases, by Observation 19, we are done. ◀

Because of Lemma 20, to prove Theorem 11, we can focus on FVS on clique-grid graphs, where the input also contains a $\mathcal{O}(\sqrt{k})$-NCTD. That is, the input of FVS on clique-grid graphs is a tuple $(G, f, k, \mathcal{T})$ where G is a clique-grid graph with representation f and $\mathcal{T} = (T, \beta)$ is a $\mathcal{O}(\sqrt{k})$-NCTD of G. Notice that if there is a cell (i, j) of f, such that $|f^{-1}(i, j)| \geq k + 3$, then there is no feedback vertex set of size at most k in G, because $f^{-1}(i, j)$ is a clique of size at least $k + 3$ in G.

▶ **Observation 21.** *Let $(G, f, k, \mathcal{T})$ be an instance of* FVS*, where G is a clique-grid graph with representation f. If there is a cell (i, j) in f such that $|f^{-1}(i, j)| \geq k+3$, then $(G, f, k, \mathcal{T})$ is a* No*-instance.*

The following observation follows from the fact that $\mathcal{T} = (T, \beta)$ is a $\mathcal{O}(\sqrt{k})$-NCTD and $|f^{-1}(i, j)| \leq k + 2$ for any cell (i, j) of f.

▶ **Observation 22.** *For any $v \in V(T)$, $|\beta(v)| = \mathcal{O}(k^{1.5})$.*

Notice that G has a feedback vertex set of size at most k if and only if there is a vertex subset $F \subseteq V(G)$ of cardinality at least $|V(G)| - k$ such that $G[F]$ is a forest. Hence, instead of stating the problem as finding a k sized feedback vertex set in G, we can state it as finding an induced subgraph H of G with maximum number of vertices such that H is a forest.

Max Induced Forest (MIF) **Parameter:** k
Input: A clique-grid graph G with representation f and an integer k such that $\mathcal{T}$ is a $c\sqrt{k}$-NCTD of G and for any cell (i, j) in f, $\lvert f^{-1}(i, j)\rvert \leq k + 2$, where c is a constant
Question: Is there subset $W \subseteq V(G)$ such that $G[W]$ is a forest and $\lvert W\rvert \geq \lvert V(G)\rvert - k$

▶ **Observation 23.** *Let $(G, f, k, \mathcal{T})$ be an instance of* MIF*. Then $(G, f, k, \mathcal{T})$ is a* Yes*-instance of* MIF *if and only if $(G, f, k, \mathcal{T})$ is a* Yes*-instance of* FVS*.*

By Lemma 20 and Observations 21 and 23, to prove Theorem 11, it is sufficient that we prove the following result (which is the focus of the rest of this section).

▶ **Lemma 24.** MIF *on clique-grid graphs can be solved in time $2^{\mathcal{O}(\sqrt{k} \log k)} \cdot n^{\mathcal{O}(1)}$.*

Proof sketch. We explain a DP algorithm which given as input $(G, f, k, \mathcal{T})$ where G is a clique-grid graph with representation f, $\mathcal{T} = (T, \beta)$ is a $c\sqrt{k}$-NCTD, c is a constant and $|f^{-1}(i, j)| \leq k + 2$ for any cell (i, j) of f and outputs Yes if there is an induced forest with at least $|V(G)| - k$ vertices and outputs No otherwise. Here we use the term solution for a vertex subset $S \subseteq V(G)$ with the property that $G[S]$ is a forest. First notice that any solution S contains at most 2 vertices from $f^{-1}(i, j)$ for any cell (i, j). Now, the following claim follows from the fact that $\mathcal{T}$ is a $c\sqrt{k}$-NCTD and any solution contain at most 2 vertices from $f^{-1}(i, j)$ for any cell (i, j).

▶ **Claim 2.** *For any $v \in V(T)$ and any solution S, $|S \cap \beta(v)| \leq 2c\sqrt{k}$.*

We first briefly explain what is the table entries in a standard DP algorithm for our problem on graphs of bounded treewidth [12]. Then we explain that in fact many of the entries we compute in the standard DP table is redundant in our case, because of Observation 22 and Claim 2. That is, Observation 22 and Claim 2, shows that only $2^{\mathcal{O}(\sqrt{k} \log k)} |V(G)|^{\mathcal{O}(1)}$ many

states in the DP table are relevant in our case. Recall that for any $v \in V(T)$, $\gamma(v)$ denote the union of the bags of v and its descendants. The standard DP table for our problem will have an entry indexed by $(v, U, U_1 \uplus U_2 \ldots U_\ell = U)$ where $v \in V(T)$, $U \subseteq \beta(v)$. The table entry $\mathcal{A}[v, U, U_1 \uplus U_2 \ldots U_\ell]$ stores the following information: the maximum cardinality of a vertex subset $W \subseteq G[\gamma(v)]$ such that $W \cap \beta(v) = U$, $G[W]$ is a forest with a set of connected components $\mathcal{C}$ and for any $C \in \mathcal{C}$, either $V(C) \cap \beta(v) = \emptyset$ or $V(C) \cap \beta(v) = U_i$ for some $i \in [\ell]$. Notice that the total number of DP table entries is bounded by $\mathsf{tw}^{\mathcal{O}(\mathsf{tw})}|V(G)|^{\mathcal{O}(1)}$ where tw is the width of the tree decomposition $\mathcal{T}$. One can easily show that the computation of the DP table at a node can be done in time polynomial in the size of the tables of its children.

By Observation 22 and Claim 2, we know that for any bag $\beta(v)$ in $\mathcal{T}$, the potential number of subsets of $\beta(v)$ which can be part of any solution is at most $2^{\mathcal{O}(\sqrt{k}\log k)}$. This implies that we only need to compute the DP table entries for indices $(v, U, U_1 \uplus U_2 \ldots U_\ell = U)$ where $v \in V(T)$, $U \subseteq \beta(v)$ and $|U| \leq 2c\sqrt{k}$. Thus, the size of DP table, and hence the time to compute it takes $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ time. This concludes the description. ◀

References

1. Jochen Alber and Jiří Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. In *Foundations of Information Technology in the Era of Network and Mobile Computing*, pages 26–37. Springer, 2002.
2. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. Assoc. Comput. Mach.*, 42(4):844–856, 1995.
3. Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, 1994.
4. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010.
5. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
6. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.
7. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.
8. Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003. doi:10.1016/S0196-6774(02)00294-8.
9. Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.
10. Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.
11. Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007. doi:10.1007/s00454-006-1273-8.
12. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
13. Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.
14. Erik D. Demaine and MohammadTaghi Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *Proceedings of the 16th Annual ACM-SIAM*

Symposium on Discrete Algorithms (SODA 2005), pages 590–601, New York, 2005. ACM-SIAM.

15 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

16 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

17 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

18 Adrian Dumitrescu and János Pach. Minimum clique partition in unit disk graphs. *Graphs and Combinatorics*, 27(3):399–411, 2011.

19 Fedor V. Fomin, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering. In *Proceedings of the 57th Annual Symposium on Foundations of Computer Science (FOCS), to appear*, 2016.

20 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of the ACM*, 63(4):29, 2016. doi:10.1145/2886094.

21 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, Hitting and Packing Cycles in Subexponential Time on Unit Disk Graphs. *ArXiv e-prints*, April 2017. URL: https://arxiv.org/abs/1704.07279.

22 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 748–759, 2011.

23 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and geometric graphs. In *SODA'12 Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete algorithms*, pages 1563–1575. SIAM, 2012.

24 F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 503–510. SIAM, 2010.

25 William K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

26 Sariel Har-Peled and Mira Lee. Weighted geometric set cover problems revisited. *JoCG*, 3(1):65–85, 2012.

27 Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294, pages 717–728. Springer, 2015.

28 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.

29 Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J. Rosenkrantz, and Richard Edwin Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

30 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

31 Hiro Ito and Masakazu Kadoshita. Tractability and intractability of problems on unit disk graphs parameterized by domain area. In *Proceedings of the 9th International Symposium on Operations Research and Its Applications (ISORA10)*, pages 120–127, 2010.

32 Bart M. P. Jansen. Polynomial kernels for hard problems on disk graphs. In *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 6139, pages 310–321. Springer, 2010.

33 Karl Kammerlander. C 900 – An advanced mobile radio telephone system with optimum frequency utilization. *IEEE journal on selected areas in communications*, 2(4):589–597, 1984.

34 Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5125 of *Lecture Notes in Computer Science*, pages 575–586, 2008.

35 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

36 Dániel Marx. Efficient approximation schemes for geometric problems? In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, volume 3669, pages 448–459. Springer, 2005.

37 Nabil H. Mustafa, Rajiv Raman, and Saurabh Ray. Settling the apx-hardness status for geometric set cover. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 541–550. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.64.

38 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems & applications. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 232–243. IEEE Computer Society, 1998.

39 Stéphan Thomassé. A quadratic kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.

40 DW Wang and Yue-Sun Kuo. A study on two geometric location problems. *Information processing letters*, 28(6):281–286, 1988.

41 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

42 Yu-Shuan Yeh, J. Wilson, and S. Schwartz. Outage probability in mobile telephony with directive antennas and macrodiversity. *IEEE journal on selected areas in communications*, 2(4):507–511, 1984.

43 Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1037–1049, 2015. doi:10.1007/978-3-662-48350-3_86.

A Polynomial-Time Randomized Reduction from Tournament Isomorphism to Tournament Asymmetry*

Pascal Schweitzer

RWTH Aachen University, Aachen, Germany
schweitzer@informatik.rwth-aachen.de

Abstract

The paper develops a new technique to extract a characteristic subset from a random source that repeatedly samples from a set of elements. Here a characteristic subset is a set that when containing an element contains all elements that have the same probability.

With this technique at hand the paper looks at the special case of the tournament isomorphism problem that stands in the way towards a polynomial-time algorithm for the graph isomorphism problem. Noting that there is a reduction from the automorphism (asymmetry) problem to the isomorphism problem, a reduction in the other direction is nevertheless not known and remains a thorny open problem.

Applying the new technique, we develop a randomized polynomial-time Turing-reduction from the tournament isomorphism problem to the tournament automorphism problem. This is the first such reduction for any kind of combinatorial object not known to have a polynomial-time solvable isomorphism problem.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures, F.1.3 Reducibility and Completeness

Keywords and phrases graph isomorphism, asymmetry, tournaments, randomized reductions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.66

1 Introduction

The graph automorphism problem asks whether a given input graph has a non-trivial automorphism. In other words the task is to decide whether a given graph is asymmetric. This computational problem is typically seen in the context of the graph isomorphism problem, which is itself equivalent under polynomial-time Turing reductions to the problem of computing a generating set for all automorphisms of a graph [17]. As a special case of the latter, the graph automorphism problem obviously reduces to the graph isomorphism problem. However, no reduction from the graph isomorphism to the graph automorphism problem is known. In fact, while many computational problems surrounding structural equivalence of combinatorial objects can all be Turing-reduced to one another, the relationship between the graph automorphism and the graph isomorphism problem remains a repeatedly posed open question (see for example [1, 2, 12, 14]).

With Babai's new ground-breaking algorithm [7] that solves the graph isomorphism problem and thereby also the graph automorphism problem in quasi-polynomial time, the question arises whether it is possible to go further and devise a polynomial-time algorithm.

* See [23], http://arxiv.org/abs/1704.08529, for the full version of the paper including the missing proofs.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 66; pp. 66:1–66:14

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

For such an endeavor to succeed, special cases such as the group isomorphism and the tournament isomorphism problem, for which the currently fastest algorithms have a running time of $n^{O(\log n)}$, should also be solvable in polynomial time. Tournaments, which are graphs in which between every pair of vertices there exists exactly one directed edge, also have an automorphism problem associated with them, asking whether a given tournament is asymmetric[1]. Again, for this problem the currently best running time is $n^{O(\log n)}$ and analogously to general graphs there is a simple reduction from the automorphism problem to the isomorphism problem, but no reverse reduction has been known.

In this paper we show that there is a randomized polynomial-time Turing reduction from the tournament isomorphism problem to the tournament automorphism problem. This is the first such reduction for any kind of combinatorial object (apart from polynomial-time solvable cases of course).

The main new technical tool that we develop in the first part of the paper is a technique to exploit an oracle to the graph automorphism problem in order to obtain a non-trivial automorphism-invariant partition of a graph that is finer than the orbit partition (Sections 2–5). We call the parts of such a partition suborbits. This technique is essentially applicable to all graph classes, not just tournaments. It hinges on a method to extract a characteristic subset from a random source that repeatedly samples from a set of elements. Here we say that a set is characteristic if it is a union of level sets of the probability function.

In the second part of the paper we show that, for tournaments, access to suborbits suffices to compute automorphism groups (Section 6). For this we adapt the group-theoretic divide and conquer approach of Luks [16] to our situation. We exploit that automorphism groups of tournaments are solvable and we leave it as an open question whether something similar can be forged that is applicable to the group isomorphism problem (see Section 7).

It might be worth noting that the techniques actually do not use any of the new structural insights from the quasi-polynomial-time algorithm of [7]. Rather, the randomized sampling idea is heavily based on an older practical randomized algorithm designed to quickly detect non-isomorphism ([15, 21]). It appears to be one of the few cases where randomization helps to derive a theoretical result for an isomorphism problem. We also borrow some ideas from a paper of Arvind, Das, and Mukhopadhyay concerned with tournament canonization [4].

The necessity for randomization to obtain theoretical results in the context of isomorphism checking appears to be quite rare. The earliest result exploiting randomization seems to go to back to Babai [5] and is a randomized algorithm for checking isomorphism of graphs of bounded color class size. However that algorithm is actually a Las Vegas algorithm (an algorithm that does not make errors), and in the meantime deterministic algorithms are available [11]. However, for the new reduction in this paper it seems unclear how to remove the use of randomization and even how to remove the possibility for errors.

Related work

With respect to related work, we focus on results concerning graph automorphism as well as results concerning tournaments and refer the reader to other texts (for example [6, 7, 14, 18,

[1] Many publications in the context of graph isomorphism use the term rigid graph. However, the literature is inconsistent on the notion of a rigid graph, which can for example refer to having no non-trivial automorphism or no non-trivial endomorphism. We will use the notion asymmetric, which only ever means the former. Furthermore, we suggest the name graph asymmetry problem over graph automorphism problem, so as not to confuse it with the computational problem to compute the automorphism group.

22]) for a general introduction to the graph isomorphism problem, current algorithms and overviews over complexity theoretic results.

Tournament automorphism. Let us start by highlighting two results specifically concerned with the tournament automorphism problem. Arvind, Das, and Mukhopadhyay [4] show that if tournament isomorphism is polynomial-time solvable then tournament canonization can be reduced in polynomial time to canonization of asymmetric tournaments. This implies now, with the result of the current paper, that from a canonization algorithm for asymmetric tournaments we can obtain a randomized canonization algorithm for tournaments in general. (In other words, the main theorem of our paper transfers to canonization.) On the hardness side, Wager [25, 27] shows that tournament automorphism is hard for various circuit complexity classes (NL, $\mathsf{C_=L}$, PL, DET, $\mathsf{MOD_kL}$) under $\mathsf{AC^0}$ reductions.

Graph automorphism A lot of information on the complexity of graph automorphism can be found in the book by Köbler, Schöning, and Torán [14]. Concerning hardness of the automorphism problem, improving previous results of Torán [24], Wagner shows hardness results for graphs of bounded maximum degree [26, 27]. Agrawal and Arvind show truth table equivalence of several problems related to graph automorphism [1] and Arvind, Beigel, and Lozano study modular versions of graph automorphism [3] which for $k \in \mathbb{N}$ ask whether the number of automorphisms of a given graph is divisible by k.

The graph automorphism problem is of interest in quantum computing since it can be encoded as a hidden shift problem, as opposed to the graph isomorphism problem that is only known to be encodable as a hidden subgroup problem [10, 13].

Recently, Allender, Grochow, and Moore [2] developed a zero-error randomized reduction from graph automorphism to MKTP, the problem of minimizing time-bounded Kolmogorov complexity, a variant of the minimum circuit size problem. In that paper they also extend this to a bounded-error randomized reduction from graph isomorphism to MKTP.

Tournament isomorphism. Concerning the tournament isomorphism problem, the currently fastest algorithm [8] has a running time of $n^{O(\log n)}$. With respect to hardness, Wagner's results for tournament automorphism also apply to tournament isomorphism [25].

Ponomarenko showed that isomorphism of cyclic tournaments can be decided in polynomial time [19], where a cyclic tournament is a tournament that has an automorphism that is a permutation with a single cycle spanning all vertices. Furthermore he showed that isomorphism of Schurian tournaments can be decided in polynomial time [20].

2 Sampling characteristic subsets

Let M be a finite set. We define a *sampler* S over M to be a probability measure $\mathrm{Pr_S}\colon M \to [0,1]$ on the elements of M. We think of a sampler as an oracle that we can invoke in order to obtain an element of M. That is, given a sampler, we can sample a sequence of elements $m_1, \ldots, m_t$ where each m_i is sampled independently from M according to $\mathrm{Pr_S}$.

We call a subset M' of M *characteristic* with respect to S if for all $m, m' \in M$ it holds that $m \in M'$ and $\mathrm{Pr_S}(m') = \mathrm{Pr_S}(m)$ implies $m' \in M'$. Another way of formulating this condition is that M' is invariant under all probability-preserving bijections $\varphi\colon M \to M$, that is, those bijections that satisfy $\mathrm{Pr_S}(m) = \mathrm{Pr_S}(\varphi(m))$ for all $m \in M$.

When considering sampling algorithms we will not assume that we know the size of the set M. Our goal is to repeatedly invoke a sampler M so as to find a characteristic subset.

The main difficulty in this is that we can never precisely determine the probability $\Pr_S(m)$ of an element m. Indeed, the only thing we can hope for is to get a good estimate for such a probability. The following lemma indicates that this might be helpful since the set of probabilities cannot be arbitrarily dense.

▶ **Lemma 1.** *Let* $\Pr_S$ *be a discrete probability measure on the set* M*. Let* $P = \{\Pr_S(m) \mid m \in M\}$ *be the set of probabilities that occur. For every positive integer* i *there is a* $j \in \{6i+1, \ldots, 8i\}$ *such that* $[(j-1/4)/(8i^2), (j+1/4)/(8i^2)] \cap P = \emptyset$*.*

Using the lemma we can design an algorithm that, with high probability, succeeds at determining a characteristic set.

▶ **Theorem 2.** *There is a deterministic algorithm that, given* $\varepsilon > 0$ *and given access to a sampler* S *over an unknown set* M *of unknown size, runs in expected time polynomial in* $1/(\max_{m \in M} \Pr_S(m)) \leq |M|$ *and* $\ln 1/\varepsilon$ *and outputs a non-empty subset of* M *that is characteristic with probability* $1 - \varepsilon$*.*

The proof of the theorem makes repeated use of Chernoff bounds. The main difficulty is that $|M|$ is not known to the algorithm. The lengthy proof can be found in the full version [23].

We note several crucial observations about any algorithm solving the problem just described. There is no algorithm that for every set M and sampler S always outputs the same set M' with high probability.

Indeed, consider the set $M = \{a, b\}$. Choosing $\Pr_S(a) = \Pr_S(b) = 1/2$ means that M' must be $\{a, b\}$. Choosing $\Pr_S(a) = 1$ and $\Pr_S(b) = 0$ implies that M' must be $\{a\}$. However, there is a continuous deformation between these two samplers, while possibilities for the set M' are discrete. It is not difficult to see that the probability distribution of the output set M' must be continuous in the space of samplers, and thus, whatever the algorithm may be, there must be samplers for which the algorithm sometimes outputs $\{a\}$ and sometimes outputs $\{a, b\}$.

The theorem only establishes polynomial running time. If we are however interested in small running times, one might even ask whether it is possible to devise an algorithm running in time sublinear in $|M|$. However, recalling the coupon collector theorem and considering uniform samplers one realizes that one cannot expect to make do with $o(|M| \log |M|)$ samplings. However, if the set M is of algebraic nature, for example forms a group, then there might be meaningful ways to sample characteristic substructures (see Section 7).

3 Gadget constructions for asymmetric tournaments

There are several computational problems fundamentally related to the graph isomorphism problem. This relation manifests formally as polynomial-time Turing (or even many-one) reductions between the computational tasks. Such reductions are typically based on gadget constructions which we revisit in this section.

While the *graph isomorphism problem* GI asks whether two given graphs are isomorphic, in the search version of this decision problem an explicit isomorphism is to be found, whenever one exits. The *graph automorphism problem* GA asks whether a given graph has a non-trivial automorphism (i.e., an automorphism different from the identity). In other words the task is to decide whether the given graph is asymmetric. Two other related problems are the task AUT to determine generators for the automorphism group $\mathrm{Aut}(G)$ and the task to determine the size of the automorphism group $|\mathrm{Aut}(G)|$.

For all named problems there is a colored variant, where the given graphs are vertex colored and isomorphisms are restricted to be color preserving. We denote the respective problems by col-GI, col-GA and col-AUT.

It is well known that between all these computational problems – except GA – there are polynomial-time Turing reductions (we refer for example to [9], [14], [17]). Concerning the special case of GA, while there is a reduction from GA to the other problems, a reverse reduction is not known.

The reductions are typically stated for general graphs, but many of the techniques are readily applicable to restricted graph classes. By a *graph class* we always mean a collection of possibly directed graphs closed under isomorphism. The *isomorphism problem for graphs in* $\mathcal{C}$, denoted $\mathsf{GI}_{\mathcal{C}}$, is the computational task to decide whether two given input graphs from $\mathcal{C}$ are isomorphic. If one of the input graphs is not in $\mathcal{C}$ the answer of an algorithm may be arbitrary, in fact the algorithm may even run forever. Analogously, for each of the other computational problems that we just mentioned, we can define a problem restricted to $\mathcal{C}$ giving us for example $\mathsf{GA}_{\mathcal{C}}$, and $\mathsf{AUT}_{\mathcal{C}}$ and the colored versions $\text{col-}\mathsf{GI}_{\mathcal{C}}$, $\text{col-}\mathsf{GA}_{\mathcal{C}}$, and $\text{col-}\mathsf{AUT}_{\mathcal{C}}$.

As remarked in [4], most of the reduction results for general graphs transfer to the problems for a graph class $\mathcal{C}$ if one has, as essential tool, a reduction from $\text{col-}\mathsf{GI}_{\mathcal{C}}$ to $\mathsf{GI}_{\mathcal{C}}$.

▶ **Theorem 3** (Arvind, Das, Mukhopadhyay [4]). *Suppose that for a graph class* $\mathcal{C}$ *there is a polynomial-time many-one reduction from* $\text{col-}\mathsf{GI}_{\mathcal{C}}$ *to* $\mathsf{GI}_{\mathcal{C}}$ *(i.e.,* $\text{col-}\mathsf{GI}_{\mathcal{C}} \leq_m^p \mathsf{GI}_{\mathcal{C}}$*)*[2]*. Then*

1. $\mathsf{GA}_{\mathcal{C}}$ *polynomial-time Turing-reduces to* $\mathsf{GI}_{\mathcal{C}}$ *(i.e.,* $\mathsf{GA}_{\mathcal{C}} \leq_T^p \mathsf{GI}_{\mathcal{C}}$*),*
2. *The search version of* $\mathsf{GI}_{\mathcal{C}}$ *polynomial-time Turing-reduces to the decision version of* $\mathsf{GI}_{\mathcal{C}}$*, and*
3. $\mathsf{AUT}_{\mathcal{C}}$ *polynomial-time Turing-reduces to* $\mathsf{GI}_{\mathcal{C}}$ *(i.e.,* $\mathsf{AUT}_{\mathcal{C}} \leq_T^p \mathsf{GI}_{\mathcal{C}}$*).*

In this paper we are mainly interested in two classes of directed graphs, namely the class of tournaments Tour and the class of asymmetric tournaments AsymTour. For the former graph class, a reduction from the colored isomorphism problem to the uncolored isomorphism problem is given in [4].

▶ **Theorem 4** (Arvind, Das, Mukhopadhyay [4]). *The colored tournament isomorphism problem is polynomial-time many-one reducible to the (uncolored) tournament isomorphism problem (i.e.,* $\text{col-}\mathsf{GI}_{\mathsf{Tour}} \leq_m^p \mathsf{GI}_{\mathsf{Tour}}$*).*

However, for our purposes we also need the equivalent statement for asymmetric tournaments. Taking a closer look at the reduction described in [4] yields the desired result. In fact it also shows that the colored asymmetry problem reduces to the uncolored asymmetry problem. Denoting for a graph class $\mathcal{C}$ by $\mathsf{Asym}\mathcal{C}$ the class of those graphs in $\mathcal{C}$ that are asymmetric (i.e., have a trivial automorphism group), we obtain the following.

▶ **Lemma 5.**

1. *The isomorphism problem of colored asymmetric tournaments is polynomial-time many-one reducible to the isomorphism problem for (uncolored) asymmetric tournaments (i.e.,* $\text{col-}\mathsf{GI}_{\mathsf{AsymTour}} \leq_m^p \mathsf{GI}_{\mathsf{AsymTour}}$*).*
2. *The colored tournament asymmetry problem is polynomial-time many-one reducible to the (uncolored) tournament asymmetry problem (i.e.,* $\text{col-}\mathsf{GA}_{\mathsf{Tour}} \leq_m^p \mathsf{GA}_{\mathsf{Tour}}$*).*

[2] Let us remark for completeness that a Turing reduction assumption $\text{col-}\mathsf{GI}_{\mathcal{C}} \leq_T^p \mathsf{GI}_{\mathcal{C}}$ actually suffices for the theorem.

As mentioned above, reductions for computational problems on general graphs can often be transferred to the equivalent problems restricted to a graph class $\mathcal{C}$. However, let us highlight a particular reduction where this is not the case. Indeed, it is not clear how to transfer the reduction from GI to AUT (which involves taking unions of graphs) to a reduction from $\mathsf{GI}_{\mathcal{C}}$ to $\mathsf{AUT}_{\mathcal{C}}$, even when provided a reduction of $\mathsf{col\text{-}GA}_{\mathcal{C}}$ to $\mathsf{GI}_{\mathcal{C}}$. For the class of tournaments however, we can find such a reduction, of which we can make further use.

▶ **Lemma 6.**

1. *The isomorphism problem for tournaments polynomial-time Turing-reduces to the task to compute a generating set for the automorphism group of a tournament (i.e.,* $\mathsf{col\text{-}GI}_{\mathsf{Tour}} \leq_T^p \mathsf{AUT}_{\mathsf{Tour}}$*).*
2. *The isomorphism problem for colored asymmetric tournaments is polynomial-time many-one reducible to tournament asymmetry (i.e.,* $\mathsf{col\text{-}GI}_{\mathsf{AsymTour}} \leq_m^p \mathsf{GA}_{\mathsf{Tour}}$*).*
3. *The search version of the isomorphism problem for colored asymmetric tournaments Turing-reduces to tournament asymmetry.*

Proof. Suppose we are given two tournaments T_1 and T_2 on the same number of vertices n for which isomorphism is to be decided. By Theorem 4 we can assume that the tournaments are uncolored. Let $\mathrm{Tri}(T_1, T_2)$ be the tournament obtained by forming the disjoint union of the three tournaments T_1, T_1' and T_2 where $T_1 \cong T_1'$. We add edges from all vertices of T_1 to all vertices of T_1', from all vertices of T_1' to all vertices of T_2 and from all vertices of T_2 to all vertices of T_1. We observe that two vertices that are contained in the same of the three sets $V(T_1)$, $V(T_2)$, $V(T_1')$ have n common out-neighbors. However, two vertices that are not contained in the same of these three sets have at most $n-1$ common out-neighbors. We conclude that an automorphism of $\mathrm{Tri}(T_1, T_2)$ preserves the partition of $V(\mathrm{Tri}(T_1, T_2))$ into the three sets $V(T_1)$, $V(T_1')$ and $V(T_2)$.

Given a generating set for $\mathrm{Aut}(\mathrm{Tri}(T_1, T_2))$ it holds that there is some generator that maps a vertex from $V(T_1)$ to a vertex from $V(T_2)$ if and only if T_1 and T_2 are isomorphic. This proves the first part of the lemma.

Suppose additionally that T_1 and T_2 are asymmetric. We then further conclude that the tournament $\mathrm{Tri}(T_1, T_2)$ has a non-trivial automorphism if and only if T_1 and T_2 are isomorphic. This shows that the decision version of asymmetric tournament isomorphism reduces to tournament asymmetry. Since the search version is Turing-reducible to the decision version of isomorphism (Theorem 3) this finishes the proof. ◀

For Turing reductions, the converse of the previous lemma also holds. In fact the converse holds for arbitrary graph classes. The first part of this converse is a well known techniques that goes back to Mathon [17].

▶ **Lemma 7.** *Let $\mathcal{C}$ be a graph class.*

1. *The task to compute a generating set for the automorphism group of graphs in $\mathcal{C}$ Turing-reduces to the isomorphism problem for colored graphs in $\mathcal{C}$ (i.e.,* $\mathsf{AUT}_{\mathcal{C}} \leq_T^p \mathsf{col\text{-}GI}_{\mathcal{C}}$*).*
2. *Asymmetry checking for graphs in $\mathcal{C}$ polynomial-time Turing-reduces to isomorphism checking of asymmetric colored graphs in $\mathcal{C}$ (i.e.,* $\mathsf{GA}_{\mathcal{C}} \leq_T^p \mathsf{col\text{-}GI}_{\mathsf{Asym}\mathcal{C}}$*).*

4 Invariant automorphism samplers from asymmetry tests

As discussed before, the asymmetry problem of a class of graphs reduces to the isomorphism problem of graphs in this class. However, whether there is a reduction in the reverse, or whether the asymmetry problem may actually be computationally easier than the isomorphism

Algorithm 1 An invariant automorphism sampler for tournaments using an asymmetry oracle.

Input: A tournament T that is not asymmetric and an oracle O for tournament asymmetry.
Output: An automorphism $\varphi \in \mathrm{Aut}(T) \setminus \{\mathrm{id}\}$. As a random variable, the outputs of the algorithm form an invariant automorphism sampler for T.

$T_{\mathrm{next}} \leftarrow T$
while $\mathrm{Aut}(T_{\mathrm{next}}) \neq \{\mathrm{id}\}$ **do**
  Pick a vertex v independently, uniformly at random among all non-singleton color classes in T_{next}.
  $T \leftarrow T_{\mathrm{next}}$
  $T_{\mathrm{next}} \leftarrow T_{(v)}$ // individualize v
end while // at this point T_{next} is asymmetric
Let V' be the set of those vertices that have the same color in T as v.
Let V'' be the set of those vertices v'' in $V' \setminus \{v\}$ for which $\mathrm{Aut}(T_{(v'')}) = \{\mathrm{id}\}$.
Let V''' be the set of those vertices v''' in V'' for which $T_{\mathrm{next}} \cong T_{(v''')}$.
// use Part 2 of Lemma 6
Pick a vertex $u \in V'''$ uniformly at random.
Compute an isomorphism φ from T_{next} to $T_{(u)}$. // there is only one such isomorphism
return φ

problem is not known. To approach this question, we now explore what computational power we could get from having available an oracle for the asymmetry problem.

An *invariant automorphism sampler for a graph* G is a sampler over $\mathrm{Aut}(G) \setminus \{\mathrm{id}\}$ which satisfies the property that if $\Pr_{\mathrm{S}}(\varphi) = p$ then $\Pr_{\mathrm{S}}(\psi^{-1} \circ \varphi \circ \psi) = p$ for all $\psi \in \mathrm{Aut}(G)$. We first show how to use an oracle for asymmetry to design an invariant automorphism sampler for a tournament T.

▶ **Lemma 8.** *Given an oracle for asymmetry of tournaments* ($\mathsf{GA}_{\mathsf{Tour}}$) *we can construct for every given colored (or uncolored) tournament* T *that is not asymmetric an invariant automorphism sampler. The computation time (and thus the number of oracle calls) required to sample once from* S *is polynomial in* $|V(T)|$.

Proof. Let O_1 be an oracle for uncolored tournament asymmetry. By Lemma 5, we can transform the oracle O_1 for the asymmetry of uncolored tournaments into an oracle O_2 for asymmetry of colored tournaments. By Lemma 6 Part 2, we can also assume that we have an oracle O_3 that decides the isomorphism problem of colored asymmetric tournaments. More strongly, Lemma 6 Part 3 makes a remark on the search version, thus we can assume that O_3 also solves the isomorphism search problem for asymmetric tournaments.

To obtain the desired sampler S we proceed as follows. In the given tournament T we repeatedly fix (by individualization, i.e., giving it a special color) uniformly, independently at random more and more vertices until the resulting tournament is asymmetric. This gives us a sequence of colored tournaments $T = T_0, T_1, \ldots, T_t$ such that $\mathrm{Aut}(T_t) = \{\mathrm{id}\}$, $\mathrm{Aut}(T_{t-1}) \neq \{\mathrm{id}\}$ and such that $T_t = (T_{t-1})_{(v)}$ for some vertex v. In other words, T_t is obtained from T_{t-1} by individualizing v which makes the graph asymmetric. Using the available oracle O_2, we can compute the set V'' of those vertices v'' in $V(T) \setminus \{v\}$ that have the same color as v such that $\mathrm{Aut}((T_{t-1})_{(v'')}) = \{\mathrm{id}\}$. There must be at least one vertex in V'' since T_{t-1} is not asymmetric. Using the oracle O_3, we can then compute the subset $V''' \subseteq V''$ of those vertices v''' for which $(T_{t-1})_{(v''')}$ and T_t are isomorphic. Next, we pick a vertex $u \in V'''$

uniformly at random. Since both $(T_{t-1})_{(u)}$ and T_t are asymmetric, using the oracle O_3 for the isomorphism search problem we can compute an isomorphism φ from $(T_{t-1})_{(u)}$ to T_t. This isomorphism φ is unique and it is a non-trivial automorphism of $\mathrm{Aut}(T)$. Algorithm 1 gives further details.

Invariance. he invariance follows directly from the fact that all steps of the algorithm either consist of choosing a vertex uniformly at random or computing an object that is invariant with respect to all automorphisms fixing all vertices that have been randomly chosen up to this point.

Running time. Concerning the running time, one call of Algorithm 1 uses less than $2n$ calls to oracle O_2 and at most n calls to oracle O_3. The overall running time is thus polynomial. ◀

Let us comment on whether the technique of the lemma can be applied to graph classes other than tournaments. For the technique to apply to a graph class $\mathcal{C}$, we require the oracle O_2, which solves colored asymmetry $\mathcal{C}$, and the oracle O_3 which solves the isomorphism search problem for asymmetric colored objects in $\mathcal{C}$. (The oracle O_1 is a special case of O_2.) In the case of tournaments, having an oracle O_1 (i.e., an oracle for uncolored asymmetry) is sufficient to simulate the oracles O_2 and O_3, but this is not necessarily possible for all graph classes $\mathcal{C}$. It is however possible to simulate such oracles for every graph class that satisfy some suitable (mild) assumptions, as can be seen from the discussion in Section 3. In particular, given an oracle for asymmetry of all graphs we can construct an invariant automorphism sampler for all graphs that are not asymmetric.

5 Invariant suborbits from invariant automorphism samplers

Let G be a directed graph. Let S be an invariant automorphism sampler for G. We now describe an algorithm that, given access to an asymmetry oracle, constructs a non-discrete partition of $V(G)$ which is finer than or at least as fine as the orbit partition of G under $\mathrm{Aut}(G)$ and invariant under $\mathrm{Aut}(G)$. Here, a partition π is invariant under $\mathrm{Aut}(G)$ if $\pi = \psi(\pi)$ for all $\psi \in \mathrm{Aut}(G)$. (A partition is discrete if it consists only of singletons.)

▶ **Theorem 9.** *For every $c \in \mathbb{N}$, there is a randomized polynomial-time algorithm that, given a graph G and an invariant automorphism sampler* S *for G constructs with error probability at most $\frac{1}{|G|^c}$ a non-discrete partition π of $V(G)$ such that*
1. *π is finer than or at least as fine as the orbit partition of $V(G)$ under* $\mathrm{Aut}(G)$ *and*
2. *π is invariant under* $\mathrm{Aut}(G)$.

The algorithm also provides a set of certificates $\Phi = \{\varphi_1, \ldots, \varphi_m\} \subseteq \mathrm{Aut}(G)$ such that for every pair of vertices $v, v' \in V(G)$ that lie in the same class of π there is some φ_i with $\varphi_i(v) = v'$.

Proof. Let $M = \{(v, w) \mid v, w \in V(G), v \neq w, \exists \varphi \in \mathrm{Aut}(G) \colon \varphi(v) = w\}$ be the set of pairs of two distinct vertices lying in the same orbit. With the sampler S we can simulate a sampler S′ over M invariant under $\mathrm{Aut}(G)$ as follows. To create an element for S′ we sample an element φ from S and uniformly at random choose an element v from the support $\mathrm{supp}(\varphi) = \{x \in V(G) \mid \varphi(x) \neq x\}$ of φ. Then the element for S′ is $(v, \varphi(v))$. It follows form the construction that S′ is a sampler for M. Moreover, since all random choices are independent and uniform, S′ is invariant under automorphisms.

Using the algorithm from Theorem 2 we can thus compute a characteristic subset M' of M. Since S′ is $\mathrm{Aut}(G)$-invariant, the fact that M' is characteristic implies that it is also

$\mathrm{Aut}(G)$-invariant. For the given $c \in \mathbb{N}$, to obtain the right error bound, we choose ε to be $\frac{1}{|G|^c}$ for the algorithm from Theorem 2. Then the error probability is at most $\varepsilon = \frac{1}{|G|^c}$ and the running time is polynomial in $|M| = O(|G|^2)$ and $\ln |G|^c = O(|G|)$ and thus polynomial in the size of the graph.

Regarding M' as a binary relation on $V(G)$ we compute the transitive closure and let π be the partition of $V(G)$ into equivalence classes of said closure, where vertices that do not appear at all as entries in M' form their own class. By construction, elements that are in the same class of π are in the same orbit under $\mathrm{Aut}(G)$. Moreover π is $\mathrm{Aut}(G)$-invariant since M' is $\mathrm{Aut}(G)$-invariant.

To provide certificates for the elements in M' we store all elements given to us by S. For each $(v, w) \in M'$ we can thus compute an automorphism of $\varphi_{v,w} \in \mathrm{Aut}(G)$ with $\varphi_{v,w}(v) = w$. For pairs in the transitive closure of M' we then multiply suitable automorphisms. ◀

If a partition π satisfies the conclusion of the lemma, we call it an *invariant collection of suborbits*. We call the elements of Φ the *certificates*. Let us caution the reader that the set Φ returned by the algorithm is not necessarily characteristic. Moreover, the orbits of the elements in Φ might not necessarily be contained within classes of π. An *oracle for invariant suborbits* returns (π, Φ), where for asymmetric inputs π is discrete and $\Phi = \{\mathrm{id}\}$.

6 Computing the automorphism group from invariant suborbits

To exploit invariant suborbits we make use of the powerful group-theoretic technique to compute stabilizer subgroups.

▶ **Theorem 10** (Luks [16]). *There is an algorithm that, given a permutation group Γ on $\{1, \ldots, n\}$ and subset $B \subseteq \{1, \ldots, n\}$, computes (generators for) the setwise stabilizer of B. If Γ is solvable, then this algorithm runs in polynomial time.*

We will apply the theorem in the following form: Let G be a graph and Γ a solvable permutation group on $V(G)$. Then $\Gamma \cap \mathrm{Aut}(G)$ can be computed in polynomial time. This follows directly from the theorem by considering the induced action of Γ on pairs of vertices from $V(G)$ and noting that $\Gamma \cap \mathrm{Aut}(G)$ consists of those elements that stabilize the edge set.

In our algorithm we will also use the concept of a quotient tournament (that can for example implicitly be found in [4], see also [22]). Let T be a tournament and let π be a partition of $V(T)$ in which all parts have odd size. We define T/π, the *quotient of T modulo π*, to be the tournament on π (i.e., the vertices of T/π are the parts of π) where for distinct $C, C' \in V(T/\pi) = \pi$ there is an edge from C to C' if and only if in T there are more edges going from C to C' than edges going from C' to C. Note that since both $|C|$ and $|C'|$ are odd there are either more edges going from C to C' or more edges going from C' to C. This implies that T/π is a tournament.

▶ **Theorem 11.** *Suppose we are given as an oracle a randomized Las Vegas algorithm that computes invariant suborbits for tournaments in polynomial time. Then we can compute the automorphism group of tournaments in polynomial time.*

Proof. We describe an algorithm that computes the automorphism group of a colored tournament given a randomized oracle that provides invariant suborbits.

Algorithm 2 Computing the automorphism of a tournament using invariant suborbits.

Input: A (colored) tournament T and an oracle O for invariant suborbits with certificates.
Output: A generating set for the automorphism group $\mathrm{Aut}(T)$.

if T is not monochromatic **then** // Case 0
  Let Col be the set of vertex colors of T.
  for $c \in COL$ **do**
    Let V^c be the set vertices in T of color c.
    $\Psi^c \leftarrow \mathrm{Aut}(T[V^c])$ // recursion
    Let $\widehat{\Psi^c}$ be the set of extensions of Ψ^c to $V(T)$ obtained by fixing vertices outside V^c.
  end for
  $\Psi = \bigcup_{c \in \mathrm{Col}} \widehat{\Psi^c}$
  return $\langle \Psi \rangle \cap \mathrm{Aut}(T)$ // solvable group stabilizer
end if
$(\pi, \Phi) \leftarrow O(T)$ // π forms invariant suborbits of T, Φ the set of certificates
if π is discrete **then** // T is asymmetric
  return $\{\mathrm{id}\}$
else if $\pi = \{V(T)\}$ **then** // Case 1
  Choose $v \in V(T)$ arbitrarily.
  Let T' be obtained from T by coloring v with 1, all in-neighbors of v with 2 and other vertices with 3.
  return $\Phi \cup \mathrm{Aut}(V(T'))$ // recursion
else if $\exists C, C' \in \pi \colon |C| \neq |C'|$ **then** // Case 2
  Let T' be obtained from T by coloring each vertex v with color $|[v]_\pi|$.
  return $\mathrm{Aut}(V(T'))$ // recursion
else // Case 3
  For $C \in \pi$ we let T_C be the graph obtained from $T[C]$ by picking an arbitrary vertex $v \in C$ and coloring v with 1, all in-neighbors of v with 2 and other vertices with 3.
  for $\{(C, C') \in \pi \mid C \neq C'\}$ **do**
    Compute $\mathrm{Aut}(\mathrm{Tri}(T_C, T'_C))$ and extract an isomorphism $\varphi_{(C,C')} \colon T[C] \to T[C']$ whenever such an isomorphism exists. // recursion
  end for
  if $\exists C, C' \in \pi \colon T[C] \ncong T[C']$ **then** // Case 3a
    Let T' be obtained from T by coloring $V(T)$ so that v and v' have the same color if and only if $T[([v])] \cong T[([v'])]$.
    return $\mathrm{Aut}(T')$ // recursion
  else // Case 3b
    $\Psi \leftarrow \mathrm{Aut}(T/\pi)$ // recursion on the quotient
    $\widehat{\Psi} \leftarrow \{\widehat{g} \mid g \in \Psi\}$, where $\widehat{g}(v) = \varphi_{([v], g([v]))}(v)$.
    for $\{C \in \pi\}$ **do**
      $\Upsilon_C \leftarrow \mathrm{Aut}(T[C])$ // recursion
      Compute $\widehat{\Upsilon}_C$ the lifts of elements in Υ_C by fixing vertices outside C.
    end for
    return $\langle \widehat{\Psi} \cup \bigcup_{C \in \pi} \widehat{\Upsilon}_C \rangle \cap \mathrm{Aut}(T)$ // solvable group stabilizer
  end if
end if

Description of the algorithm. Let T be a given colored tournament.

Case 0: T is not monochromatic. If T is not monochromatic then we proceed as follows:
Let Col be the set of colors that appear in T. For $c \in \mathrm{Col}$, let V^c be the set of vertices of color c and let $T^c = T[V^c]$ be the subtournament induced by the vertices in V^c.
We recursively compute $\mathrm{Aut}(T^c)$ for all $c \in \mathrm{Col}$. Let Ψ^c be the set of generators obtained as an answer. We lift every generator to a permutation of $V(T)$ by fixing all vertices outside of V^c. Let $\widehat{\Psi^c}$ be the set of lifted generators of Ψ^c and let $\Psi = \bigcup_{c\in\mathrm{Col}} \widehat{\Psi^c}$ be the set of all lifted generators. Since $\mathrm{Aut}(T^c) = \langle \Psi^c \rangle$ is solvable, we conclude that $\langle \Psi \rangle$ is a direct product of solvable groups and thus solvable. We can thus compute $\langle \Psi \rangle \cap \mathrm{Aut}(T)$ using Theorem 10 and return the answer.
This concludes Case 0.
In every other case we first compute a partition π into suborbits using the oracle and a corresponding set of certificates Φ. For a partition π of some set V we denote for $v \in V$ by $[v]_\pi$ the element of π containing v. We may drop the index when it is obvious from the context. If $|T| = 1$ then we simply return the identity.

Case 1: π is trivial. In case π is trivial (i.e., $\pi = \{V(T)\}$), we know that T is transitive. We choose an arbitrary vertex $v \in V(T)$. Let λ be the coloring of $V(T)$ satisfying $\lambda(u) = 1$ if $u = v$, $\lambda(u) = 2$ if $(u, v) \in E(T)$, and $\lambda(u) = 3$ otherwise. We recursively compute a generating set Ψ for $\mathrm{Aut}(T')$, where T' is T recolored with λ. We then return $\Psi \cup \Phi$.

Case 2: not all classes of π have the same size. We color every vertex with the size of the class of π in which it is contained. Now T is not monochromatic anymore and we recursively compute $\mathrm{Aut}(T)$ with T having said coloring. (In other words, we proceed as in Case 0.)

Case 3: all classes of π have the same size but π is non-trivial. We compute for each pair of distinct equivalence classes C and C' of π an isomorphism $\varphi_{(C,C')}$ from $T[C]$ to $T[C']$ or determine that no such isomorphism exists, as follows: We choose for each C an arbitrary vertex $v \in C$. We let T_C be the tournament obtained from $T[C]$ by coloring v with 1, all in-neighbors of v with 2 and other vertices with 3. We let $T_{C,C'} = \mathrm{Tri}(T_C, T_{C'})$ be the triangle tournament of T_C and $T_{C'}$ where $(T_C)'$ is an isomorphic copy of T_C (as defined in Section 3 in the proof of Lemma 6).
Using recursion we compute $\mathrm{Aut}(T_{C,C'})$. From the result we can extract an isomorphism from $T[C]$ to $T[C']$ since $V(T[C])$ and $V(T[C'])$ are blocks of $T_{C,C'}$.
Case 3a: If it is not the case that for every pair C, C' of color classes there is an isomorphism from $T[C]$ to $T[C']$ then we color the vertices of T so that v, v' have the same color if and only if there is an isomorphism from $T[([v])]$ to $T[([v'])]$, where as before for every vertex u we denote by $[u]$ the class of π containing u. With this coloring, T is not monochromatic anymore and we recursively compute $\mathrm{Aut}(T)$ with T having said coloring. (In other words, we proceed as in Case 0.)
Case 3b: Otherwise, for every pair C, C' of color classes, there is an isomorphism from $T[C]$ to $T[C']$. Note that all color classes are of odd size since $T[C]$ is transitive (as dictated by π). Thus, we can compute the quotient tournament T/π. We recursively compute a generating set $\Psi = \{g_1, \ldots, g_t\}$ for the automorphism group of T/π.
We lift each g_i to a permutation $\widehat{g_i}$ of $V(T)$ as follows. The permutation $\widehat{g_i}$ maps each vertex v to $\varphi_{([v],g_i([v]))}(v)$. Since g_i is a permutation and each $\varphi_{(C,C')}$ is a bijection, the map $\widehat{g_i}$ is a permutation of $V(T)$. Let $\widehat{\Psi} = \{\widehat{g_1}, \ldots, \widehat{g_t}\}$ be the set of lifted generators.
As next step, for each class C we recursively compute a generating set Υ_C for $\mathrm{Aut}(T[C])$. We lift each generator in Υ_C to a permutation of $V(T)$ by fixing all vertices outside of C obtaining the set $\widehat{\Upsilon}_C$ of lifted generators.

Consider the group Γ generated by the set $\widehat{\Psi} \cup \bigcup_{C \in \pi} \widehat{\Upsilon}_C$. As a last step, using Theorem 10 we compute the subgroup $\Gamma' = \Gamma \cap \mathrm{Aut}(T)$.

The details of this algorithm are given in Algorithm 2. For the running time analysis and the correctness argument of the algorithm we refer to the full version of the paper [23]. ◀

We have now assembled all the required parts to prove the main theorem of the paper.

▶ **Corollary 12.**

1. *There is a randomized (one-sided error) polynomial-time Turing reduction from tournament isomorphism to asymmetry testing of tournaments (i.e.,* $\mathsf{GI}_{\mathsf{Tour}} \leq^p_{r,T} \mathsf{GA}_{\mathsf{Tour}}$*).*
2. *There is a randomized polynomial-time Turing reduction from the computational task to compute generators of the automorphism group of a tournament to asymmetry testing of tournaments (i.e.,* $\mathsf{AUT}_{\mathsf{Tour}} \leq^p_{r,T} \mathsf{GA}_{\mathsf{Tour}}$*).*

Proof. Recall that a two-sided error algorithm for an isomorphism search problem can be readily turned into a one-sided error algorithm by checking the output isomorphism for correctness. Thus, by Lemma 6 Part 1 it suffices to prove the second part of the corollary.

Combining Lemma 8 and Theorem 9, from an oracle to tournament asymmetry we obtain a randomized Monte Carlo (i.e., with possible errors) algorithm that computes invariant suborbits. Given a Las Vegas algorithm (i.e., no errors) for suborbits, the previous theorem provides us with a computation of the automorphism group of tournaments.

It remains to discuss the error probability we get from using a Monte Carlo algorithm instead of a Las Vegas algorithm. Since there is only a polynomial number of oracle calls, and since the error bound in Theorem 9 can be chosen smaller than $\frac{1}{|G|^c}$ for every fixed constant c, the overall error can be chosen to be arbitrarily small. ◀

7 Discussion and open problems

This paper is concerned with the relationship between the asymmetry problem $\mathsf{GA}_{\mathcal{C}}$ and isomorphism problem $\mathsf{GI}_{\mathcal{C}}$. While under mild assumptions there is a reduction from the former to the latter, a reduction in the other direction is usually not known. However, for tournaments we now have such a randomized reduction.

The first question that comes to mind is whether the technique described in this paper applies to other graph classes. While the sampling techniques from Sections 2 to 5 can be applied to all graph classes that satisfy mild assumptions (e.g., $\mathsf{col\text{-}GI}_{\mathcal{C}} \leq^p_t \mathsf{GI}_{\mathcal{C}}$ and $\mathsf{col\text{-}GI}_{\mathsf{Asym}\mathcal{C}} \leq^p_t \mathsf{GI}_{\mathsf{Asym}\mathcal{C}}$) the algorithm described in Section 6 crucially uses the fact that automorphism groups of tournaments are solvable. This is not the case for general graphs, so for the open question of whether GI reduces to GA this may dampen our enthusiasm. However, what may bring our enthusiasm back up is that there are key classes of combinatorial objects that share properties similar to what we need.

In particular, this brings us to the question whether the techniques of the paper can be applied to group isomorphism. Just like for tournament isomorphism, finding a faster algorithm for group isomorphism (given by multiplication table) is a bottleneck for improving the run-time bound for isomorphism of general graphs beyond quasi-polynomial. Since outer-automorphism groups of simple groups are solvable, we ask: Can we reduce the group isomorphism problem to the isomorphism problem for asymmetric groups? This question is significant since an asymmetry assumption on groups is typically a strong structural property and may help to solve the entire group isomorphism problem. However, here one has to be careful to find the right notion of asymmetry since all groups have inner automorphisms. For such notions different possibilities come to mind.

A second natural open question would be whether there is a deterministic version of the algorithms given in this paper.

As a last open problem recall that it was shown in Section 2 that one can extract a characteristic subset for a sampler over a set M in time that depends polynomially on M. Since the automorphism group of a graph can be superpolynomial in the size of the graph, we had to take a detour via suborbits in Section 5. There can be no general way to extract a characteristic subset of M in polynomial time if $|M|$ is not polynomially bounded, since we might never see an element twice. However, if M has an algebraic structure, in particular if M is a permutation group over a polynomial size set, this is not clear. Thus we ask: Is there a polynomial-time (randomized) algorithm that extracts a characteristic subgroup using a sampler Γ over a permutation group?

References

1 Manindra Agrawal and Vikraman Arvind. A note on decision versus search for graph automorphism. *Inf. Comput.*, 131(2):179–189, 1996. doi:10.1006/inco.1996.0097.

2 Eric Allender, Joshua A. Grochow, and Cristopher Moore. Graph isomorphism and circuit size. *CoRR*, abs/1511.08189, 2015. URL: http://arxiv.org/abs/1511.08189.

3 Vikraman Arvind, Richard Beigel, and Antoni Lozano. The complexity of modular graph automorphism. *SIAM J. Comput.*, 30(4):1299–1320, 2000. doi:10.1137/S0097539799358227.

4 Vikraman Arvind, Bireswar Das, and Partha Mukhopadhyay. Isomorphism and canonization of tournaments and hypertournaments. *J. Comput. Syst. Sci.*, 76(7):509–523, 2010. doi:10.1016/j.jcss.2009.09.001.

5 László Babai. Monte carlo algorithms in graph isomorphism testing. Technical Report 79-10, Université de Montréal, 1979.

6 László Babai. Automorphism groups, isomorphism, reconstruction. In *Handbook of combinatorics, Vol. 2*, pages 1447–1540. Elsevier, Amsterdam, 1995.

7 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

8 László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC 1983, Boston, Massachusetts, USA, 25-27 April*, pages 171–183. ACM, 1983. doi:10.1145/800061.808746.

9 Kellogg S. Booth and Charles J. Colbourn. Problems polynomially equivalent to graph isomorphism. Technical Report CS-77-04, Comp. Sci. Dep., Univ. Waterloo, 1979.

10 Andrew M. Childs and Pawel Wocjan. On the quantum hardness of solving isomorphism problems as nonabelian hidden shift problems. *Quantum Information & Computation*, 7(5):504–521, 2007.

11 Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October, FOCS 1980*, pages 36–41. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.34.

12 Sumanta Ghosh and Piyush P. Kurur. Permutation groups and the graph isomorphism problem. In *Perspectives in Computational Complexity: The Somenath Biswas Anniversary Volume*, pages 183–202. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-05446-9_11.

13 Sean Hallgren, Alexander Russell, and Amnon Ta-Shma. The hidden subgroup problem and quantum computation using group representations. *SIAM J. Comput.*, 32(4):916–934, 2003. doi:10.1137/S009753970139450X.

14 Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity*. Progress in Theoretical Computer Science. Birkhäuser Boston, Inc., Boston, MA, 1993. doi:10.1007/978-1-4612-0333-9.

15 Martin Kutz and Pascal Schweitzer. Screwbox: a randomized certifying graph-non-isomorphism algorithm. In *Proceedings of the Nine Workshop on Algorithm Engineering and Experiments, ALENEX 2007, New Orleans, Louisiana, USA, January 6, 2007*. SIAM, 2007. doi:10.1137/1.9781611972870.14.

16 Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982. doi:10.1016/0022-0000(82)90009-5.

17 Rudolf Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979. doi:10.1016/0020-0190(79)90004-8.

18 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

19 I. N. Ponomarenko. Polynomial time recognition and testing of isomorphism of cyclic tournaments. *Journal of Mathematical Sciences*, 70(4):1890–1911, 1994. doi:10.1007/BF02112430.

20 I. N. Ponomarenko. Bases of schurian antisymmetric coherent configurations and an isomorphism test for schurian tournaments. *Journal of Mathematical Sciences*, 192(3):316–338, 2013. doi:10.1007/s10958-013-1398-2.

21 Pascal Schweitzer. *Problems of unknown complexity: Graph isomorphism and Ramsey theoretic numbers*. PhD thesis, Universität des Saarlandes, Germany, 2009.

22 Pascal Schweitzer. Towards an isomorphism dichotomy for hereditary graph classes. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 689–702. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.689.

23 Pascal Schweitzer. A polynomial-time randomized reduction from tournament isomorphism to tournament asymmetry. *CoRR*, abs/1704.08529, 2017. full version of the paper. URL: http://arxiv.org/abs/1704.08529.

24 Jacobo Torán. On the hardness of graph isomorphism. *SIAM J. Comput.*, 33(5):1093–1108, 2004. doi:10.1137/S009753970241096X.

25 Fabian Wagner. Hardness results for tournament isomorphism and automorphism. In *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 572–583. Springer, 2007. doi:10.1007/978-3-540-74456-6_51.

26 Fabian Wagner. Hardness results for isomorphism and automorphism of bounded valence graphs. In *SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19-25, 2008, Volume II*, pages 131–140. Safarik University, Kosice, Slovakia, 2008.

27 Fabian Wagner. *On the Complexity of Isomorphism Testing for Restricted Classes of Graphs*. PhD thesis, Universität Ulm, Germany, 2010.

A $(1+\epsilon)$-Approximation for Unsplittable Flow on a Path in Fixed-Parameter Running Time*

Andreas Wiese

Department of Industrial Engineering and Center for Mathematical Modeling, Universidad de Chile, Santiago, Chile
awiese@dii.uchile.cl

Abstract

Unsplittable Flow on a Path (UFP) is a well-studied problem. It arises in many different settings such as bandwidth allocation, scheduling, and caching. We are given a path with capacities on the edges and a set of tasks, each of them is described by a start and an end vertex and a demand. The goal is to select as many tasks as possible such that the demand of the selected tasks using each edge does not exceed the capacity of this edge. The problem admits a QPTAS and the best known polynomial time result is a $(2+\epsilon)$-approximation. As we prove in this paper, the problem is intractable for fixed-parameter algorithms since it is W[1]-hard. A PTAS seems difficult to construct. However, we show that if we combine the paradigms of approximation algorithms and fixed-parameter tractability we can break the mentioned boundaries. We show that on instances with $|OPT| = k$ we can compute a $(1+\epsilon)$-approximation in time $2^{O(k \log k)} n^{O_\epsilon(1)} \log u_{\max}$ (where $u_{\max}$ is the maximum edge capacity). To obtain this algorithm we develop new insights for UFP and enrich a recent dynamic programming framework for the problem. Our results yield a PTAS for (unweighted) UFP instances where $|OPT|$ is at most $O(\log n / \log \log n)$ and they imply that the problem does not admit an EPTAS, unless W[1] = FPT.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Combinatorial optimization, Approximation algorithms, Fixed-parameter algorithms, Unsplittable Flow on a Path

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.67

1 Introduction

The Unsplittable Flow on a Path problem is motivated by many different settings such as scheduling, bandwidth allocation, and caching. We are given an undirected path $G = (V, E)$ with a capacity $u(e) \in \mathbb{N}$ for each edge $e \in E$. Also, we are given a set of n tasks T. Each task $i \in T$ is specified by a subpath $P(i) \subseteq V$ between (and including) the start (i. e., leftmost) vertex $s(i) \in V$ and the end (i. e., rightmost) vertex $t(i) \in V$, and a demand $d(i) \in \mathbb{N}$. For instance, the tasks can be seen as jobs with start and end times that need some portion of a shared resource. The goal is to select a subset $T' \subseteq T$ of tasks of maximum total size such that for each edge e the total demand of the selected tasks using e does not exceed $u(e)$.

UFP is NP-hard [7, 14] and therefore approximation algorithms have been studied for the problem. The best known polynomial time algorithm yields a $(2+\epsilon)$-approximation [3] (improving previous results [5, 7]) and for some cases even a $(1+\epsilon)$-approximation is known [6, 17, 12]. Also, there is a QPTAS [4, 6] which makes it plausible that also a PTAS

* This work was partially supported by the Millennium Nucleus Information and Coordination in Networks ICM/FIC RC130003.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 67; pp. 67:1–67:13


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

exists. However, despite the recent progress on the problem [6, 17] the best known polynomial time result is still the mentioned $(2+\epsilon)$-approximation [3], and a PTAS seems difficult to construct. We note that all above algorithms even work in the weighted case of the problem in which each task has a profit associated with it and one wants to maximize the total profit of the selected tasks. However, as it is typical in the FPT-literature, in this paper we restrict ourselves to the unweighted case, i.e., we assume that each task yields a profit of one. No better results than the above are known for this case.

Another approach for NP-hard problems are fixed-parameter algorithms. For any instance one identifies a parameter k, e.g., the value of the optimal solution, and searches for an exact algorithm with a running time of $f(k) \cdot n^{O(1)}$ for some (typically exponential) function f. A problem is called fixed-parameter tractable (FPT) if it admits such an algorithm. We refer the reader to the recent textbook by Cygan et al. [13] for an introduction to FPT algorithms. For UFP, throughout this paper our parameter will be the size of the optimal solution. Unfortunately, as we show in this paper, it is unlikely that UFP is FPT since the problem is W[1]-hard.

1.1 Our Contribution

In this paper we show that if we combine the paradigms of approximation and fixed-parameter algorithms then we can break the mentioned barriers of $2+\epsilon$ and W[1]-hardness for UFP. We present an algorithm with a running time of $2^{O(k \log k)} n^{O_\epsilon(1)} \log u_{\max}$ that computes a $(1+\epsilon)$-approximation for any instance with $|OPT| = k$, i.e., the computed solution contains at least $k/(1+\epsilon)$ tasks, where $u_{\max} = \max_e u(e)$. Hence, we obtain a PTAS for (unweighted) UFP for instances where $|OPT| \leq O(\log n / \log \log n)$.

We first consider the special case where the number of different task demands in the input is bounded by a parameter k'. We show that then there exists an optimal solution that has a special structure. We can guess this structure in FPT-time (i.e., the number of possibilities is bounded by a function $f(k, k')$) and based on this we can construct the solution deterministically.

Then, we generalize this result to the setting where the tasks have arbitrary demands and the edge capacities are in a bounded range. There, we show that if we have k tasks with relatively small demand (of at most a $1/k$-fraction of the capacities of the edges they are using) then they form a feasible solution and we are done. Otherwise, in FPT-time we can guess which of these tasks are contained in the optimal solution and then focus on the remaining, relatively large tasks. For those we use a result from [6] that shows that there is a $(1+\epsilon)$-approximative solution in which (essentially) each edge has some slack that equals the minimum size of a large task. Thus, there is still a near-optimal solution if we round up the task demands so that they have only $f(k)$ many different demands. On the resulting instance, we apply our FPT-algorithm from above.

To obtain an algorithm for the general case with arbitrary task demands and edge capacities we use the machinery that was introduced in [17] in order to turn a PTAS for UFP with resource augmentation (i.e., where the edge capacities are increased by a factor $1+\epsilon$) to a PTAS without resource augmentation. In order to apply it in our setting, we need several new ideas.

First, we prove that we can identify at most k vertices of the input path such that each input task uses one of them (for instances in which no k such vertices exist we can find a solution with k tasks using a greedy algorithm). These vertices divide the path into $k+1$ *segments*. Our algorithm proceeds in phases and in each phase we process some set of tasks. These tasks are divided into tiny and non-tiny tasks. A crucial difficulty is to

estimate how much capacity should be given to each of these groups on each edge. We cannot afford to guess this for each edge separately. However, we show that there exists a $(1+\epsilon)$-approximative solution in which this allocation has a structure that we can guess in FPT-time. For each segment S we can essentially argue that (i) either it is not used by non-tiny tasks and in this case we can give the whole edge capacity to the tiny tasks (ii) or all tiny tasks use the same capacity on each edge of S which we can guess. Then, for the remaining decisions for the tiny tasks we call the FPT-algorithm for the case of a bounded range of edge capacities. For the non-tiny tasks we know that each segment is used by at most $O_\epsilon(1)$ of them and we can guess them step by step in polynomial time.

Finally, we prove that UFP is W[1]-hard (if parametrized by the size of the optimal solution) which makes it unlikely that there is a fixed-parameter algorithm for it that computes an optimal solution, instead of an approximation. Also, this implies that UFP does not admit an EPTAS (i.e., an $(1+\epsilon)$-approximation algorithm with a running time of $f(\epsilon)\cdot n^{O(1)}$ for some function f), unless W[1] = FPT.

We hope that our new techniques yield progress for eventually finding a PTAS for UFP. For instance, many algorithms for UFP are based on a recursive decomposition of the problem, embedded into a DP [3, 6, 17]. Using our new algorithm we can now stop such a recursion once the optimal solution of a considered subproblem has a size of at most $O(\log n/\log\log n)$. Also, for the setting of bounded edge capacities we proved that there exist optimal solutions with a special structure (inherited from the case of few different task demands). This insight might be useful beyond our result. Note that even for uniform edge capacities no PTAS is known for UFP.

We would like to point out that in the literature there exists the notion of an FPT-approximation scheme (FPT-AS) which is a $(1+\epsilon)$-approximation algorithm with a running time of $f(\epsilon,k)\cdot n^{O(1)}$ for some suitable function f, while our algorithm has a running time of $2^{O(k\log k)}n^{O_\epsilon(1)}\log u_{\max}$ and thus ϵ appears in the exponent of n. For UFP, we cannot hope for an FPT-AS since otherwise we could choose e.g., $\epsilon := 1/(2k)$ and obtain an FPT-algorithm for UFP, thus contradicting that the problem is W[1]-hard. There are many FPT-ASs known in the literature, see [18] and references therein.

We note that due to space constraints many proofs and details are omitted in this extended abstract.

1.2 Other related work

If all input tasks of a UFP instance have (relatively) small demand compared to the capacities of the edges they use, Chekuri et al. [12] proved that there is a $(1+\epsilon)$-approximation via LP-rounding. This unifies (and improves) previously known results for the special cases of uniform edge capacities [8] and the no-bottleneck-assumption (NBA) [9] which requires that $\max_{i\in T} d(i) \leq \min_{e\in E} u(e)$. Under the latter assumption, tasks with relatively large demands can be handled via dynamic programming, and thus $O(1)$- and $(2+\epsilon)$-approximation algorithms were known for these cases [8, 9, 12] and later such algorithms were also found for the general case of the problem [7, 3]. Another line of research on UFP is to find good LP-relaxations for the problem.

The natural LP-relaxation suffers from an integrality gap of $\Omega(n)$ [9] but with additional constraints Chekuri et al. [11] reduced it to $O(\log^2 n)$ (which was later improved to $O(\log n)$ by the same authors [10]). Anagnostopoulos et al. [2] found a compact LP for the cardinality case of UFP with constant integrality gap and an extended formulation with a constant gap for the weighted case. Grandoni et al. [16] prove the currently best integrality gap for an LP-relaxation without additional variables of $O(\log n/\log\log n)$.

1.3 Preliminaries and Notation

In a UFP instance, for each edge $e \in E$, let $T_e \subseteq T$ be the subset of tasks i using edge e, i. e., with $e \in P(i)$. For every set of tasks T' we define $d(T') := \sum_{i \in T'} d(i)$. The goal of (unweighted) UFP is to select set of tasks T' with maximum cardinality $|T'|$ such that $d(T' \cap T_e) \leq u(e)$ for each edge e. For a given instance of UFP we denote by OPT an optimal solution. Without loss of generality, we may assume that $|V| = 2n$, that each vertex is either the start-vertex or the end-vertex of exactly one input task, and that each task alone yields a feasible solution [4]. Throughout this paper, we use the notation $O_\epsilon(f(n))$ for functions that are in $O(f(n))$ if ϵ is a constant. In particular, $O_\epsilon(1)$ represents a value that depends only on ϵ.

2 Bounded task demands or edge capacities

In this section we first present an algorithm for the special case that the number of different task demands in the input instance is bounded by a parameter k'. Afterwards we will use it as a subroutine for the case where the range of edge capacities is bounded by a parameter k'' (without a bound on the task demands).

2.1 Bounded number of task demands

Suppose we are given an instance with at most k' different task demands where $|OPT| = k$. We present an algorithm with a running time of $f(k, k') \cdot n^{O(1)}$ that computes an optimal solution.

We first guess some properties of OPT. We can assume w.l.o.g. that OPT does not contain any task i such that there is a task $i' \in T \setminus OPT$ with $d(i) = d(i')$ and $P(i') \subseteq P(i)$ (otherwise we could replace i by i'). Assume that $OPT = \{i(1), ..., i(k)\}$ such that $s(i(\ell))$ lies on the left of $s(i(\ell'))$ if and only if $\ell < \ell'$. We use color-coding (see [1]) to split the input tasks into k pair-wise disjoint groups $T^1, ..., T^k$ such that for each ℓ the group T^ℓ contains $i(\ell)$. Note that in [1] the authors present a version of the color-coding method that does not require randomization.

▶ **Lemma 1** (implied by [1]). *By increasing the running time by a factor $2^{O(k)} \log n$ we can assume that the input tasks are colored with k colors $\{1, ..., k\}$ such that for each $\ell \in [k]$ the task $i(\ell)$ is colored with color ℓ.*

Next, we guess for each $\ell \in \{1, ..., k\}$ the demand of the task $i(\ell)$. Since for each task there are k' possibilities, the total number of guesses is $(k')^k$. We remove from T^ℓ all tasks whose demand does not equal the demand that we guessed for $i(\ell)$. Furthermore, we remove from T^ℓ each task i such that there is another task $i' \in T^\ell$ with $i \neq i'$ and $P(i') \subseteq P(i)$. Note that by our assumption about OPT above this does not remove the task $i(\ell)$. Denote by $\bar{T}^\ell$ the resulting set for each $\ell \in [k]$.

In the next lemma we define an (optimal) solution OPT' with k tasks. It will turn out that the information guessed so far is sufficient to construct OPT'. We say that a task $i \in T$ is *compatible* with a set of tasks T' if $T' \cup \{i\}$ is a feasible solution.

▶ **Lemma 2.** *There is a solution $OPT' = \{i'(1), ..., i'(k)\}$ (with $|OPT'| = k$) that satisfies for each $\ell \in \{1, ..., k\}$ that the task $i'(\ell)$ is the task in $\bar{T}^\ell$ with the leftmost start vertex that is compatible with the tasks $i'(1), ..., i'(\ell - 1)$.*

Proof. We prove the lemma by transforming $OPT =: OPT_0$ step by step into OPT'. For each $\ell \in \{1, ..., k\}$ let OPT_ℓ be the solution obtained after ℓ steps. We will ensure that each OPT_ℓ is feasible and contains k tasks. Let $i'(1)$ be the task in $\bar{T}^1$ with the leftmost start vertex. If $i'(1) \in OPT_0$ then we define $OPT_1 := OPT$. If $i'(1) \notin OPT_0$ then we replace $i(1)$ by $i'(1)$ and we define $OPT_1 := OPT \setminus \{i(1)\} \cup \{i'(1)\}$. We claim that OPT_1 is feasible. To this end, let us first consider all edges on the left of $s(i(1))$. By definition of $i(1)$, there is no task in OPT starting on the left of $s(i(1))$. By assumption, the task $i'(1)$ alone yields a feasible solution. Thus, for each edge e on the left of $s(i(1))$ we have that $d(OPT_1 \cap T_e) \leq u(e)$. By construction of the set $\bar{T}^1$ we know that $P(i(1)) \not\subseteq P(i'(1))$ and thus $t(i'(1))$ lies on the left of $t(i(1))$. Since $d(i(1)) = d(i'(1))$ and OPT is feasible we have that $d(OPT_1 \cap T_e) \leq u(e')$ for each edge e' on the right of $s(i(1))$.

Assume by induction that we constructed a feasible solution $OPT_\ell = \{i'(1), ..., i'(\ell), i(\ell + 1), ..., i(k)\}$ such that for each $\ell' \in \{1, ..., \ell\}$ the task $i'(\ell')$ is the task in $\bar{T}^{\ell}$ with the leftmost start vertex that is compatible with the tasks $i'(1), ..., i'(\ell' - 1)$ and that OPT_ℓ contains k tasks. Let $i'(\ell + 1)$ be the task in $\bar{T}^{\ell+1}$ with the leftmost start vertex that is compatible with the tasks $i'(1), ..., i'(\ell)$. Define $OPT_{\ell+1} := OPT_\ell \setminus \{i(\ell + 1)\} \cup \{i'(\ell + 1)\}$. Clearly, $OPT_{\ell+1}$ contains k tasks. We claim that $OPT_{\ell+1}$ is feasible. Let e be an edge on the left of $s(i(\ell + 1))$. Then e is not used by the tasks $i(\ell + 1), ..., i(k)$. By definition, $i'(\ell + 1)$ is compatible with the tasks $i'(1), ..., i'(\ell')$ and thus $d(OPT_{\ell+1} \cap T_e) \leq u(e)$. Let e' be an edge on the right of $s(i(\ell + 1))$. Again, by construction of the set $\bar{T}^{\ell+1}$ we know that $t(i'(\ell + 1))$ lies on the left of $t(i(\ell + 1))$. Thus, if e' is used by $i'(\ell + 1)$ then it is also used by $i(\ell + 1)$. Since by induction OPT_ℓ is feasible, this implies that also $OPT_{\ell+1}$ is feasible. ◀

Note that the start vertices of $i'(1), ..., i'(k)$ are not necessarily ordered, i.e., it could be that $s(i'(\ell))$ lies on the right of $s(i'(\ell + 1))$. Nevertheless, due to Lemma 2 we can use now the following algorithm to find a solution of size k. We define $i'(1)$ to be the task in $\bar{T}^1$ with the leftmost start vertex. Then, for each $\ell \in \{2, ..., k\}$ we inductively define $i'(\ell)$ to be the task in $\bar{T}^{\ell}$ with the leftmost start vertex that is compatible with the tasks $i'(1), ..., i'(\ell - 1)$. This yields the solution OPT' due to Lemma 2.

▶ **Theorem 3.** *Suppose we are given an UFP instance with k' different task demands in the input. Then there is an algorithm that computes a solution of size k in time $(k \cdot k')^k n^{O(1)}$ if such a solution exists.*

2.2 FPT-range of edge capacities

We give now a $(1 + \epsilon)$-approximation algorithm with a running time of $f(k, k'') \cdot n^{O_\epsilon(1)}$ for the case that the edge capacities differ by some parameter k''. Here, we allow arbitrary task demands in the input and thus lift the assumption from the previous section. Formally, our algorithm outputs a solution of size at least $k/(1 + \epsilon)$ or asserts that there is no solution of size k.

Let $u_{\min} = \min_{e \in E} u(e)$ and $u_{\max} = \max_{e \in E} u(e)$. We assume that $u_{\max} \leq k'' \cdot u_{\min}$ where k'' is a parameter. For each task i we define its bottleneck capacity $b(i) := \min_{e \in P(i)} u(e)$. We define a task i to be *large* if $d(i) \geq b(i)/k$ and small otherwise. The next lemma shows that if there are at least k small tasks then any k of them will form a feasible solution (and hence we are done). It holds even for arbitrary edge capacities.

▶ **Lemma 4.** *Any set of at most k small tasks forms a feasible solution.*

Proof. Let T' be a set of k small tasks. We want to prove that T' is a feasible solution. Let e be an edge. We have that $d(T' \cap T_e) = \sum_{i \in T' \cap T_e} d(i) < \sum_{i \in T' \cap T_e} b(i)/k \leq \frac{1}{k} \sum_{i \in T' \cap T_e} u(e) \leq u(e)$. ◀

If there are at least k small tasks in the input then we output k of them and we are done. Otherwise, denote by OPT_S the small tasks from OPT. We guess in time 2^{k-1} the set OPT_S, select all these tasks for our final solution, and discard all other small tasks. We focus on the large tasks now. Denote by OPT_L the set of large tasks in OPT.

We borrow an idea from [6, Lemma 2.6] to achieve the following: we sacrifice a factor of $1+O(\epsilon)$ in the objective and remove some tasks from OPT_L such that if an edge e is used by at least $1/\epsilon$ tasks in OPT_L we remove at least one task from $T_e \cap OPT_L$.

▶ **Lemma 5** ([6]). *There is a set $\overline{OPT}_L \subseteq OPT_L$ with $|\overline{OPT}_L| \leq O(\epsilon) \cdot |OPT_L|$ such that for each edge e with $|T_e \cap OPT_L| \geq 1/\epsilon$ we have that $|T_e \cap \overline{OPT}_L| \geq 1$.*

We define $OPT' := OPT \setminus \overline{OPT}_L$ with $\overline{OPT}_L$ being defined as in Lemma 5. Assume for a moment that each edge is used by at least one task in $\overline{OPT}_L$. Then we know that $d(T_e \cap OPT') \leq u(e) - \min_{i \in OPT_L} d(i)$. Since all tasks in OPT_L are large we have that $\min_{i \in OPT_L} d(i) \geq \frac{1}{k} \cdot u_{\min}$. On the other hand, $|OPT'| \leq k$. Thus, OPT' remains feasible if we increase the demand of each large task to the next higher integral multiple of $\frac{1}{k^2} u_{\min}$. Since $d(i) \leq u_{\max}$ for each task $i \in T$, this yields an instance with only $\frac{u_{\max}}{\frac{1}{k^2} \cdot u_{\min}} \leq k^2 \cdot k''$ different demands. We can then apply the exact FPT-algorithm from Section 2.1 on the resulting instance.

This procedure fails if there are edges not used by tasks in $\overline{OPT}_L$. Denote those edges by E_f. However, such edges are used by only few tasks in OPT_L, at most $1/\epsilon$ many. Thus, we can employ a dynamic program (DP) that guesses those edges step by step and guess their corresponding tasks. Any two consecutive edges e_L, e_R in E_f yield a subproblem for which (like above) we can increase the demands of the tasks whose path lies strictly between e_L and e_R and then invoke the exact FPT-algorithm from Section 2.1 as a subroutine.

▶ **Theorem 6.** *There is a $(1+\epsilon)$-approximation algorithm with a running time of $(k \cdot k'')^{O(k)} n^{O(1/\epsilon)}$ for UFP-instances with $|OPT| = k$ in which the edge capacities lie within a factor k''.*

3 General case

In this section we present our main result. For any $\epsilon > 0$ and any $k \in \mathbb{N}$ we present an algorithm with a running time of $2^{O(k \log k)} \cdot n^{O_\epsilon(1)}$ that computes a solution consisting of at least $k/(1+\epsilon)$ tasks on any instance with $|OPT| = k$ or asserts that there is no solution of size k. We will assume for the moment that the input numbers are bounded by a polynomial in the input size and later explain how to lift this assumption. Our argumentation consists of the following steps:

- In Section 3.1 we use techniques from [17] in order to gain some slack (i.e., unused capacity) on the edges while losing only a factor of $1+\epsilon$ in the objective. Then we classify edges into types and supertypes according to their respective amount of slack. We do a similar classification into types/supertypes for the tasks.
- Afterwards in Section 3.2, we identify a set $\bar{V}$ of k vertices such that each input task uses at least one of them. They split the input path into $k+1$ segments for which we establish some structural properties.
- We present the main algorithm in Section 3.3, described as a (possibly exponential time) recursion. It processes the tasks in phases with one phase for each supertype.
- Finally, we embed our recursive algorithm into a dynamic program with the claimed running time and lift the assumption that the input numbers are polynomially bounded.

3.1 Classification of tasks and edges

In this subsection, we apply the machinery presented in [17] in order to classify edges via how much unused capacity (i.e., *slack*) they have in some near-optimal solution. Also, we group tasks into tiny, medium, and huge tasks. Several times we apply some standard shifting arguments to ensure that we lose at most a factor $1 + O(\epsilon)$ in the process.

▶ **Lemma 7** ([17]). *Let $\epsilon > 0$ be a constant. Given a UFP instance with optimal solution OPT, there exists a feasible solution OPT' with $|OPT'| \geq (1 - O(\epsilon)) \cdot |OPT|$ such that for each edge e there is a value $\delta_e \geq 0$ satisfying the following conditions:*

1. *either $\delta_e = (1/\epsilon^2)^j$ for some integer $j \geq 0$, or $\delta_e = 0$;*
2. *$d(T_e \cap OPT') \leq u(e) - \delta_e$;*
3. *there are at most $1/\epsilon^5$ tasks $i \in T_e \cap OPT'$ such that $d(i) \geq \epsilon^2 \cdot \delta_e$;*
4. *the total demand of all tasks $i \in T_e \cap OPT'$ such that $d(i) < \epsilon^2 \cdot \delta_e$ is at most $5\delta_e/\epsilon^3$.*

In our reasoning, we will aim at computing a solution with nearly as many tasks as OPT'. Like in [17] we group the edges and the input tasks according to the amount of slack (i.e., the δ_e-values) that they have/that the edges on their respective paths have.

For each edge $e \in E$, we define its *type* type(e) as follows: If $\delta_e = (1/\epsilon^2)^j$ for some $j \in \mathbb{N}$, then define type$(e) := j$; and if $\delta_e = 0$, then define type$(e) := -1$. We denote by $E^{(j)}$ the set of edges of type j in E. We say that a task $i \in T$ is of *type* j if $P(i)$ uses an edge of type j and no edge of type $j-1$ or lower. Let $T^{(j)} \subseteq T$ denote all tasks of type j. We write type(i) to denote the type of task i.

▶ **Definition 8.** A task $i \in T$ of type j is *huge* if $d(i) \geq \epsilon^2 \cdot \delta^{(j)}$.

Next, we group the tasks into supertypes. Each supertype consists of $1/\epsilon - 1$ (usual) types. We remove the tasks of all types $a + \ell/\epsilon - 1$ with $\ell \in \mathbb{N}$ for some offset $a \in \{0, ..., 1/\epsilon - 1\}$ and define the tasks supertypes $\mathcal{T}^{(\ell)} := \bigcup_{\ell'=a+\ell/\epsilon}^{a+\ell/\epsilon+1/\epsilon-2} T^{(\ell')}$, one for each $\ell \in \mathbb{Z}$. For a task i we say that i is of *supertype* ℓ if $i \in \mathcal{T}^{(\ell)}$ and we write stype$(i) = \ell$. Similarly, we define for the edges the supertypes $\mathcal{E}^{(\ell)} := \bigcup_{\ell'=a+\ell/\epsilon}^{a+\ell/\epsilon+1/\epsilon-2} E^{(\ell')}$ (for the same offset a as above) and write stype$(e) := \ell$ if $e \in \mathcal{E}^{(\ell)}$. This implies that edges of supertype ℓ have slacks in the range $[\left(\frac{1}{\epsilon^2}\right)^{a+\ell/\epsilon}, \left(\frac{1}{\epsilon^2}\right)^{a+\ell/\epsilon+1/\epsilon-2}] =: [s^{(\ell)}_{\min}, s^{(\ell)}_{\max}]$. The following proposition follows from a simple shifting argument.

▶ **Proposition 9.** *There exists an offset $a \in \{0, ..., 1/\epsilon - 1\}$ such that by reducing the number of tasks in OPT' by a factor $1 + O(\epsilon)$ we can assume that $OPT' \subseteq \bigcup_{\ell \in \mathbb{N}} \mathcal{T}^{(\ell)}$.*

Since we removed the tasks of all types $a + \ell/\epsilon - 1$ with $\ell \in \mathbb{N}$ we can guarantee that all non-huge tasks of a supertype $\mathcal{T}^{(\ell)}$ fit into the slack of each edge of supertype $\ell + 1$ or larger. Let $OPT'_{NH} \subseteq OPT'$ denote the tasks in OPT' that are not huge.

▶ **Lemma 10.** *Let $\mathcal{T}^{(\ell)}$ be a supertype and let $e \in \mathcal{E}^{(\ell)}$. Then $d(T_e \cap OPT'_{NH} \cap \mathcal{T}^{(\ell)}) \leq 10 \cdot \frac{1}{\epsilon^3} \cdot (\frac{1}{\epsilon^2})^{a+\ell/\epsilon+1/\epsilon-2} := d^{(\ell)}_{\max}$. Moreover, for each edge $e' \in \mathcal{E}^{(\ell')}$ with $\ell' \geq \ell + 1$ we have that $d(T_e \cap OPT'_{NH} \cap \mathcal{T}^{(\ell)}) \leq d^{(\ell)}_{\max} \leq 10\epsilon \cdot s^{(\ell+1)}_{\min} \leq 10\epsilon \cdot \delta_{e'}$.*

We split the non-huge tasks into tiny and medium tasks. Let $\mu_1, \mu_2 > 0$ with $\mu_1 < \mu_2$ be two constants to be defined later. We say that a non-huge task $i \in \mathcal{T}^{(\ell)}$ is *tiny* if $d(i) \leq \mu_1 \cdot s^{(\ell)}_{\min}$ and it is *medium* if $d(i) \geq \mu_2 \cdot s^{(\ell)}_{\min}$. Note that there are some tasks that are neither tiny nor medium, i.e., a task i with $\mu_1 \cdot s^{(\ell)}_{\min} < d(i) < \mu_2 \cdot s^{(\ell)}_{\min}$. We will neglect such tasks. Due to the following lemma, we can find values for μ_1, μ_2 such that this is justified. Also, there is a large gap between these two values which we will exploit later.

▶ **Lemma 11.** *For each $\epsilon > 0$ we can find a set of $1/\epsilon$ pairs $(\mu_1^{(1)}, \mu_2^{(1)}), ..., (\mu_1^{(1/\epsilon)}, \mu_2^{(1/\epsilon)})$ such that for one pair $(\mu_1^{(r)}, \mu_2^{(r)})$ it holds that $\mu_1 \leq \frac{\epsilon}{\alpha_\epsilon}$ with $\alpha_\epsilon := \frac{1}{\epsilon} \cdot \left(\frac{2}{\epsilon^6} + \frac{10\epsilon}{\mu_2} \cdot (1/\epsilon)^{1/\epsilon}\right)$ and the set $OPT' \cap \bigcup_\ell \{i \in \mathcal{T}^{(\ell)} | \mu_1^{(r)} \cdot s_{\min}^{(\ell)} < d(i) < \mu_2^{(r)} \cdot s_{\min}^{(\ell)}\}$ contains at most $\epsilon \cdot |OPT'|$ tasks.*

We assume that we guess the correct pair $(\mu_1^{(r)}, \mu_2^{(r)})$ and define the sets of tiny and medium tasks according to it.

3.2 Structure via segments

Next, we show that we can compute a set of at most k vertices such that the path of each input task uses one of them (otherwise we can directly find a solution with k tasks).

▶ **Lemma 12.** *In polynomial time we can identify (i) a set $\bar{V} \subseteq V$ of at most k vertices such that for each task $i \in T$ there is a vertex $v \in \bar{V}$ such that $v \in P(i)$ or (ii) a set of k tasks that form a feasible solution.*

Proof. Let i be the task with leftmost end vertex $t(i)$. We define $\bar{T} := \{i\}$ and $\bar{V} := \{t(i)\}$. We remove all tasks using $t(i)$ from the input. Note that all remaining tasks start and end on the right of $t(i)$. We iterate this process $k-1$ more times: among the remaining tasks we identify the task i' with left most end vertex $t(i')$ and we add $t(i')$ to $\bar{T}$, we add $t(i')$ to $\bar{V}$, and we remove all tasks using $t(i')$. At the end, we have that $|\bar{T}| = |\bar{V}|$ and by construction, no two tasks in $\bar{T}$ share a vertex (and thus also no edge) and each input task uses one vertex in $\bar{V}$. Hence, if $|\bar{T}| \geq k$ then we found a feasible solution with k tasks. Otherwise, the set $\bar{V}$ is the set satisfying the claim of the lemma. ◀

The vertices in $\bar{V}$ divide the path into a set of at most $k+1$ *segments* $\mathcal{S}$, i.e., any two vertices $v, v' \in \bar{V}$ such that there is no vertex of $\bar{V}$ between v and v' induce a segment $S \subseteq E$ which contains all edges between v and v'. Additionally, $\mathcal{S}$ contains a segment containing all edges between the leftmost vertex of G and the leftmost vertex in $\bar{V}$ and a segment between the rightmost vertex in $\bar{V}$ and the rightmost vertex in G. For each supertype $\mathcal{T}^{(\ell)}$ we can bound the number of huge tasks starting or ending within a segment.

▶ **Lemma 13.** *Let $\mathcal{T}^{(\ell)}$ be a supertype and let S be a segment. Then there can be at most $\frac{2}{\epsilon^6}$ huge tasks in $\mathcal{T}^{(\ell)} \cap OPT'$ that use an edge of S.*

A core problem for our algorithm is that we do not know how to allocate the edge capacities between the tiny, the medium, and the huge tasks. To this end, we prove the following lemma that will later allow us to essentially guess this allocation in FPT-time.

▶ **Lemma 14.** *By reducing the number of tasks in OPT' by at most a factor $1 + O(\epsilon)$ we can assume that for each segment $S \in \mathcal{S}$ and each supertype ℓ one of the following holds:*

- *there is no huge or medium task of supertype ℓ using any edge of S or*
- *at most α_ϵ tiny tasks of supertype ℓ start or end in S. In this case, the total demand of all tiny tasks of type j starting or ending in S is bounded by $\alpha_\epsilon \cdot \mu_1 \cdot s_{\min}^{(\ell)} \leq \epsilon \cdot s_{\min}^{(\ell)}$.*

Proof. Consider a segment S and assume that there is a huge or medium task using some edge e of S. By Lemma 13 there can be at most $\frac{2}{\epsilon^6}$ such huge tasks. Moreover, there can be at most $\frac{d_{\max}^{(\ell)}}{\mu_2 \cdot s_{\min}^{(\ell)}} \leq \frac{10\epsilon \cdot s_{\min}^{(\ell+1)}}{\mu_2 \cdot s_{\min}^{(\ell)}} \leq \frac{10\epsilon}{\mu_2} \cdot (1/\epsilon)^{1/\epsilon}$ such medium tasks and hence at most $\epsilon \cdot \alpha_\epsilon$ medium or huge tasks in total. If there are more than α_ϵ tiny tasks starting or ending in S then we remove all medium and huge tasks using an edge of S. We do this operation with

all segments S. We charge the cost of the removed huge and medium tasks to the tiny tasks. Let n' be the number of removed tasks. Then $OPT' \geq \frac{1}{2\epsilon}n'$ and thus $n' \leq 2\epsilon \cdot OPT'$. ◀

If for a segment S and a supertype ℓ the first case of Lemma 14 applies then we say that the pair (S, ℓ) is *tiny*, otherwise we say that (S, ℓ) is *huge*. As we show in the next lemma, in time FPT-time we can guess which task supertypes appear in the optimal solution.

▶ **Lemma 15.** *In time* $(\log n)^{O(k)} \leq n \cdot 2^{O(k)}$ *we can guess the set* $L = \{\ell \mid OPT \cap \mathcal{T}^{(\ell)} \neq \emptyset\}$.

Proof. Each supertype ℓ arising in the optimal solution is an integer between -1 and $\log_{1/\epsilon^2} \max_{e \in E} u(e)$. Since the input numbers are polynomially bounded this yields at most $O(\log n)$ many supertypes. For each of the k tasks in OPT' there are $O(\log n)$ options for its supertype. Thus, in time $O(\log n)^k$ we can guess all supertypes arising in OPT'. ◀

Next, we use color-coding [1] in order to guess the correct supertype of each tiny task from OPT'. More precisely, we use it in order to obtain sets $\bar{\mathcal{T}}^{(\ell)}$ for $\ell \in \mathbb{N}$ such that each tiny task $i \in OPT'$ of supertype ℓ is contained in the set $\bar{\mathcal{T}}^{(\ell)}$ (but the set $\bar{\mathcal{T}}^{(\ell)}$ possibly contains more tasks). Note that then we know the set $T \setminus \bigcup_\ell \bar{\mathcal{T}}^{(\ell)}$ which contains all medium and huge tasks in OPT'.

▶ **Lemma 16** ([1])**.** *By increasing the running time by a factor* $2^{O(k)} \cdot \log n$ *we can assume that we are given sets* $\bar{\mathcal{T}}^{(\ell)}$, $\ell \in \mathbb{N}$, *such that each tiny task* $i \in OPT'$ *of supertype* ℓ *is contained in the set* $\bar{\mathcal{T}}^{(\ell)}$.

3.3 Recursive algorithm

Denote by OPT'_T, OPT'_M, and OPT'_H the tiny, medium and huge tasks in OPT', respectively. We describe now a recursive algorithm that constructs a solution with $|OPT'|$ many tasks. We will show later how to embed it into a dynamic program that runs in FPT-time. Our algorithm proceeds in phases, each phase corresponds to one supertype ℓ. Let ℓ be the supertype of the first phase. We assume that stype$(e) \geq \ell$ for each edge e (otherwise we can reduce the instance to a set of smaller instances in which this holds).

First, in time 2^{k+1} we guess for each segment S whether (S, ℓ) is huge or tiny. For each edge $e \in S$ we allocate a certain amount of capacity $u_\ell(e)$ for the tiny tasks of supertype ℓ. A special case arises for the supertype ℓ containing the type $j = -1$. There are no tiny tasks of this supertype and we define $u_\ell(e) := 0$. Otherwise, if (S, ℓ) is huge then this means that the tiny tasks of supertype ℓ starting or ending in S have very little total capacity, at most $\epsilon \cdot s^{(\ell)}_{\min}$. However, there might be more tiny tasks that use the edges of S but do not start or end in S. Denote by x their total capacity and note that $x \leq d^{(\ell)}_{\max}$. We assign the same amount of capacity to the tiny tasks in $\bar{\mathcal{T}}^{(\ell)}$ on each edge $e \in S$. We guess the value $\bar{x} := \min\left\{d^{(\ell)}_{\max}, \left(\left\lceil x/(s^{(\ell)}_{\min} \cdot \epsilon)\right\rceil + 1\right) \cdot \epsilon \cdot s^{(\ell)}_{\min}\right\}$ and we define $u_\ell(e) := \bar{x}$ for each edge $e \in S$. There are only $O_\epsilon(1)$ many options for $\bar{x}$. Since there are at most $k+1$ segments there are only $2^{O_\epsilon(k)}$ many guesses for the huge pairs (S, ℓ).

▶ **Lemma 17.** *Let* e *be an edge of a segment* S *such that* (S, ℓ) *is huge. Then* $d(OPT'_T \cap T_e \cap \bar{\mathcal{T}}^{(\ell)}) \leq u_\ell(e)$.

Now assume that (S, ℓ) is tiny. We do not know the supertype of each edge $e \in S$. However, we know that for each edge $e \in S$ of supertype ℓ there is no huge or medium task of supertype ℓ that uses e. For each edge $e \in S$ of supertype $\ell + 1$ or larger we know that the tiny tasks of supertype ℓ use at most $d^{(\ell)}_{\max}$ units of its capacity. Therefore,

we can give to the tiny tasks of supertype ℓ the capacity of each edge $e \in S$ that is not used by the previously guessed huge tasks, up to a maximum of $d^{(\ell)}_{\max}$. We define $u_\ell(e) := \min\{u(e) - d(OPT'_{H,\geq \ell-1} \cap T_e) - (1-\epsilon)s^{(\ell)}_{\min}, d^{(\ell)}_{\max}\}$ for each edge $e \in S$ where $OPT'_{H,\geq \ell-1}$ is the set of huge tasks $i \in OPT'$ that satisfy $d(i) \geq \epsilon^2 \cdot s^{(\ell-1)}_{\min}$.

▶ **Lemma 18.** *Let ℓ be a supertype. Let e be an edge of a segment S such that (S,ℓ) is tiny. Then $d(OPT'_T \cap T_e \cap \bar{\mathcal{T}}^{(\ell)}) \leq u_\ell(e)$.*

The following lemma implies that after we assigned $u_\ell(e)$ units of capacity to the tiny tasks the remaining capacity is sufficient for the huge and medium tasks in OPT' (in particular the not yet selected ones of supertype ℓ or larger). This holds even if we assign the capacity of the tiny tasks in this manner for all supertypes $\ell' \leq \ell$.

▶ **Lemma 19.** *For each edge e of a segment S we have that $\sum_{\ell':\ell'\leq\ell} u_{\ell'}(e) + d(OPT'_H \cap T_e) + d(OPT'_M \cap T_e) \leq u(e) - \frac{1}{2} \cdot s^{(\ell)}_{\min}$ where $\ell := \text{stype}(e)$.*

For each edge e we have that $u_\ell(e) \in [\epsilon \cdot s^{(\ell)}_{\min}, d^{(\ell)}_{\max}]$, independent on whether e is in a huge or in a tiny segment. Thus, the $u_\ell(e)$ values are in a constant range. We call our FPT-algorithm for this case (see Theorem 6) with the input consisting of $\bar{\mathcal{T}}^{(\ell)}$ and the edge capacities u_ℓ. Due to Lemmas 17 and 18 it will output a solution consisting of at least $|OPT'_T \cap \bar{\mathcal{T}}^{(\ell)}|$ tasks.

Next, we want to guess the huge and medium tasks of supertype ℓ in OPT' and split the path into subpaths such that each subpath E' has the property that $\text{stype}(e') \geq \ell+1$ for each edge $e' \in E'$. First, we guess which segments S have the property that for some edge $e \in S$ we have that $\text{stype}(e) = \ell$. We can do this in time 2^{k+1}. Denote by $\mathcal{S}'$ the resulting set of segments. We process the segments in $\mathcal{S}'$ from left to right, starting with the leftmost such segment $S \in \mathcal{S}'$. We guess the leftmost and the rightmost edge in S of supertype ℓ, denote them by e_L and e_R, respectively. Then we guess the at most $O_\epsilon(1)$ medium and huge tasks of supertype ℓ that use e_L or e_R. We recurse on the subpath consisting all edges on the left of e_L. There, each edge is of supertype at least $\ell+1$. On the subpath on the right of e_R we continue splitting the remaining path into subpaths. To this end, we take the next segment $S' \in \mathcal{S}'$, guess its leftmost and rightmost edges e'_L and e'_R of supertype ℓ, and guess the $O_\epsilon(1)$ medium and huge tasks of supertype ℓ using one of them. We recurse on the subpath between e_R and e'_L, knowing that each of its edges is of supertype at least $\ell+1$. Also, there cannot be any task whose path lies strictly between e_L and e_R since each input task has to use some vertex in $\bar{V}$. We proceed with splitting the remaining segments in $\mathcal{S}'$ on the right of S'. To this end, it suffices to know e'_R and the $O_\epsilon(1)$ medium and huge tasks using it, rather than also e_L, e_R, and e'_L and the medium and huge tasks using those (apart from those that use also e'_R).

We can embed our whole recursive algorithm into a dynamic program whose running time is FPT. Here we use ideas from [17], in particular for using the slack on the edges in order to be able to "forget" some of the previously taken decisions. Crucial here is that in order to define the $u_\ell(e)$-values it is not necessary to remember all previously guessed tasks and all values $u_{\ell'}(e)$ for all $\ell' < \ell$, but only the tasks in $OPT'_{H,\geq\ell-1}$ that use the leftmost or the rightmost edge of the subpath of the respective subproblem. One can show that those can be only $O_\epsilon(1)$ many. Thus, each arising subproblem can be described by a supertype ℓ, a subpath E' of E, and the $O_\epsilon(1)$ tasks in $OPT'_{H,\geq\ell-1}$ using the leftmost or the rightmost edge of E'. This bounds the number subproblems and thus the number of DP-cells by $n^{O_\epsilon(1)} \cdot \log u_{\max}$. With an additional color coding step and some slight extensions to the above routine one can remove the assumption that the input values are polynomially bounded.

▶ **Theorem 20.** *There exists a $(1+\epsilon)$-approximation algorithm with a running time of $2^{O(k \log k)} n^{O_\epsilon(1)} \log u_{\max}$ for UFP-instances with $|OPT| = k$.*

▶ **Corollary 21.** *There is a PTAS for UFP instances that have satisfy the property that $|OPT| \leq O(\log n / \log \log n)$.*

4 W[1]-hardness

In this section we prove that UFP is W[1]-hard if the parameter k represents the number of tasks in the optimal solution.

▶ **Theorem 22.** *The unweighted Unsplittable Flow on a Path problem is W[1]-hard when parametrized by the number of tasks in the optimal solution.*

We give a reduction from the k-subset sum problem which is W[1]-hard [15]. Given a set of n values $A = \{a_1, ..., a_n\}$, a target value B and an integer k, the goal is to choose exactly k values from A that sum up to exactly B. Suppose we are given an instance of k-subset sum. First, we claim that we can assume w.l.o.g. the following properties for it.

▶ **Lemma 23.** *W.l.o.g. we can assume that there are values $\epsilon_1, ..., \epsilon_n$, not necessarily positive, such that $a_i = B/k + \epsilon_i$ for each $i \in [n]$ and that $\sum_{i=1}^{n} |\epsilon_i| < B/k$.*

We construct an instance of UFP that admits a solution with $2k$ tasks if and only if the given k-subset sum is a yes-instance. Our UFP instance has a path with $n+2$ vertices $v_0, v_1, ..., v_{n+1}$. Denote the leftmost and the rightmost edge by e_L and e_R, respectively. We define $u(e_L) = u(e_R) = B$. For all other edges e we define $u(e) := B + k \cdot \max_i |\epsilon_i|$. Assume that the values in S are ordered such that $a_1 \geq a_2 \geq ... \geq a_n$. Let $j \in [n]$. We introduce two tasks $i(j), i'(j)$ with $s(i(j)) := v_0$, $t(i(j)) := v_i$, $d(i(j)) := a_j$ $s(i'(j)) := v_i$, $t(i'(j)) = v_{n+1}$, and $d(i'(j)) := 2B/k - a_j$. See Figure 1 for a sketch.

In order to get some intuition about the constructed instance, we observe the following.

▶ **Lemma 24.** *In the constructed instance there can be no solution with more than $2k$ tasks.*

In the next lemma we show that we can construct a solution with $2k$ tasks if the given k-subset sum instance is a yes-instance: for a given set $J \subseteq [n]$ of k indices such that $\sum_{j \in J} a_j = B$ we select the tasks $i(j)$ and $i'(j)$ for each $j \in J$. One can easily verify that this yields a feasible solution.

▶ **Lemma 25.** *If the given k-subset sum instance is a yes-instance, then the constructed UFP instance has a solution with $2k$ tasks.*

Conversely, we show that if the UFP instance has a solution with $2k$ tasks then the k-subset sum instance is a yes-instance. Suppose we are given such a solution for the UFP instance. First, we establish that for each $j \in [n]$ the solution selects either both $i(j)$ and $i'(j)$ or none of these two tasks.

▶ **Lemma 26.** *Given a solution T' to the UFP instance with $2k$ tasks. Then there is a solution T'' with $2k$ tasks such that for each $j \in [n]$ we have that either $\{i(j), i'(j)\} \subseteq T''$ or $\{i(j), i'(j)\} \cap T'' = \emptyset$.*

Proof. Let j be an index such that neither $\{i(j), i'(j)\} \subseteq T'$ nor $\{i(j), i'(j)\} \cap T' = \emptyset$. First assume that $i(j) \in T'$ but $i'(j) \notin T'$. Then by construction the edge $\{v_j, v_{j+1}\}$ is used by at most $k-1$ tasks. Let j' be the smallest index greater than j such that the edge $\{v_{j'}, v_{j'+1}\}$

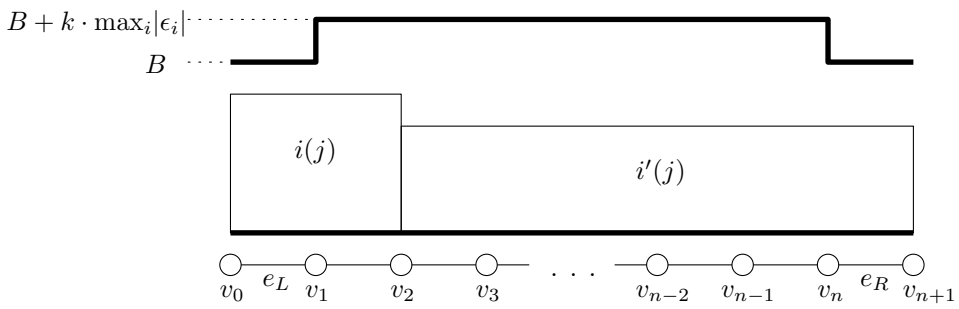


Figure 1 Sketch of the reduction used in order to prove Theorem 22. The sketch shows the tasks $i(j)$ and $i'(j)$ for only one index j.

is used by k tasks. Such an index must exist since e_R is used by k tasks from T'. Since the edge $\{v_{j'-1}, v_{j'}\}$ is used by only $k-1$ tasks this implies that $i'(j') \in T'$ but $i(j') \notin T'$. We define $\tilde{T} := T' \cup \{i'(j)\} \setminus \{i'(j')\}$. We claim that $\tilde{T}$ is feasible. The task $i'(j)$ does not use e_L and thus $\tilde{T}$ does not violate the capacity bound of e_L. Furthermore, $s(i'(j))$ lies on the left of $s(i'(j'))$ and thus $a_j \geq a_{j'}$. Hence, $d(i'(j)) = 2B/k - a_j \leq 2B/k - a_{j'} = d(i'(j'))$. Hence, $\tilde{T}$ does not violate the capacity bound of e_R. Each edge e with $e_L \neq e \neq e_R$ is used by at most k tasks. Hence, we do not violate its capacity bound (same calculation as in the proof of Lemma 25). The case that $i(j) \notin T'$ but $i'(j) \in T'$ can be handled with a similar argumentation. We repeat this process until we cannot find another index j that violates the property of the lemma. Denote by T'' the resulting set. ◀

Suppose we are given a solution T' to the UFP instance with $2k$ tasks that satisfies Lemma 26. Let J' be the set of indices j such that $i(j) \in T'$. We show in the next two lemmas that J' is a solution to the k-subset sum instance. Lemma 27 follows from our assumption that each value a_i almost equals B/k (see Lemma 23) and the fact that the edges e_L and e_R have capacity B each.

▶ **Lemma 27.** *The set T' contains exactly k tasks using e_L and exactly k tasks using e_R. Furthermore, we have that $|J'| = k$.*

▶ **Lemma 28.** *We have that $\sum_{j \in J'} a_j = B$.*

Proof. Let $T'_L \subseteq T'$ and $T'_R \subseteq T'$ denote the set of tasks in T' using e_L and e_R, respectively. Then $B = u(e_L) \geq \sum_{i \in T'_L} d(i) = \sum_{j \in J'} a_j$. On the other hand, due to Lemma 26 we have that $B = u(e_R) \geq \sum_{i \in T'_R} d(i) = \sum_{j \in J'} 2B/k - a_j$ and hence $\sum_{j \in J'} a_j \geq (\sum_{j \in J'} 2B/k) - B = B$. Therefore $\sum_{j \in J'} a_j = B$. ◀

Hence, we proved that the constructed UFP instance has a solution with $2k$ tasks if and only if the k-subset sum instance is a yes-instance. This completes the proof of Theorem 22.

▶ **Corollary 29.** *There is no EPTAS for the unweighted Unsplittable Flow on a Path problem, unless* W[1] = FPT.

Acknowledgments. The author would like to thank Tobias Mömke and Hang Zhou for helpful discussions on the topic and many comments on an earlier draft, and the anonymous referees for many helpful suggestions and comments.

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995.

2 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. Constant integrality gap LP formulations of unsplittable flow on a path. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO 2013)*, pages 25–36, 2013. doi:10.1007/978-3-642-36694-9_3.

3 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing (2+ϵ)-approximation for unsplittable flow on a path. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 26–41. SIAM, 2014.

4 Nikhil Bansal, Amit Chakrabarti, Amir Epstein, and Baruch Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC 2006)*, pages 721–729. ACM, 2006.

5 Nikhil Bansal, Zachary Friggstad, Rohit Khandekar, and Mohammad R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *Proceedings of the 20th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, pages 702–709, 2009.

6 Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 47–58, 2015. doi:10.1137/1.9781611973730.5.

7 Paul Bonsma, Jens Schulz, and Andreas Wiese. A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM Journal on Computing*, 43:767–799, 2014.

8 Gruia Călinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms*, 7:48:1–48:7, 2011. doi:10.1145/2000807.2000816.

9 Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47:53–78, 2007.

10 Chandra Chekuri, Alina Ene, and Nitish Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. Unpublished. Available at http://cs-people.bu.edu/aene/papers/ufp-full.pdf.

11 Chandra Chekuri, Alina Ene, and Nitish Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *APPROX-RANDOM 2009*, pages 42–55, 2009.

12 Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3, 2007.

13 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.

14 Andreas Darmann, Ulrich Pferschy, and Joachim Schauer. Resource allocation with time intervals. *Theoretical Computer Science*, 411:4217–4234, 2010.

15 Michael R. Fellows and Neal Koblitz. Fixed-parameter complexity and cryptography. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pages 121–131. Springer, 1993.

16 Fabrizio Grandoni, Salvatore Ingala, and Sumedha Uniyal. Improved approximation algorithms for unsplittable flow on a path with time windows. In *International Workshop on Approximation and Online Algorithms (WAOA 2015)*, pages 13–24. Springer, 2015.

17 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. To augment or not to augment: Solving unsplittable flow on a path by creating slack. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, 2017. To appear.

18 Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

Linear-Time Kernelization for Feedback Vertex Set*

Yoichi Iwata†

National Institute of Informatics, Tokyo, Japan
yiwata@nii.ac.jp

Abstract

In this paper, we give an algorithm that, given an undirected graph G of m edges and an integer k, computes a graph G' and an integer k' in $O(k^4m)$ time such that (1) the size of the graph G' is $O(k^2)$, (2) $k' \leq k$, and (3) G has a feedback vertex set of size at most k if and only if G' has a feedback vertex set of size at most k'. This is the first linear-time polynomial-size kernel for Feedback Vertex Set. The size of our kernel is $2k^2 + k$ vertices and $4k^2$ edges, which is smaller than the previous best of $4k^2$ vertices and $8k^2$ edges. Thus, we improve the size and the running time simultaneously. We note that under the assumption of NP $\not\subseteq$ coNP/poly, Feedback Vertex Set does not admit an $O(k^{2-\epsilon})$-size kernel for any $\epsilon > 0$.

Our kernel exploits *k-submodular relaxation*, which is a recently developed technique for obtaining efficient FPT algorithms for various problems. The dual of k-submodular relaxation of Feedback Vertex Set can be seen as a half-integral variant of A-path packing, and to obtain the linear-time complexity, we give an efficient augmenting-path algorithm for this problem. We believe that this combinatorial algorithm is of independent interest.

A solver based on the proposed method won first place in the 1st Parameterized Algorithms and Computational Experiments (PACE) challenge.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases FPT Algorithms, Kernelization, Path Packing, Half-integrality

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.68

1 Introduction

In the theory of parameterized complexity, we introduce parameters to problems and analyze the complexity with respect to both the input length $n = |x|$ and the parameter value k. If an algorithm runs in $f(k)n^{O(1)}$ time for any input of length n and a parameter k, it is called a *fixed-parameter tractable (FPT) algorithm*. If the $n^{O(1)}$ factor is linear, it is called a *linear-time FPT*. The typical goal of parameterized algorithms is to develop FPT algorithms with a small $f(k)$ (e.g., c^k for a small constant c) and a small $n^{O(1)}$ (e.g., linear in n). Although there are many algorithms that have been developed with smaller $f(k)$ *or* $n^{O(1)}$, achieving the smallest $f(k)$ and $n^{O(1)}$ *simultaneously* is a difficult task, and the smallest $f(k)$ factors and the smallest $n^{O(1)}$ factors are often achieved by different algorithms. Moreover, when trying to improve the $f(k)$ factor, the $n^{O(1)}$ factor is often ignored by using the O^* notation, which hides factors polynomial in n, and when trying to improve the $n^{O(1)}$ factor, the $f(k)$ factor is often ignored by assuming k is a constant.

For example, in recent papers, Iwata, Oka, and Yoshida [18], and Ramanujan and Saurabh [27] have independently obtained $O(4^km)$-time algorithms for Almost 2-SAT,

* A full version of the paper is available at https://arxiv.org/abs/1608.01463.
† Supported by JSPS KAKENHI Grant Number JP17K12643.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 68; pp. 68:1–68:14


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

which is a parameterized version of Max 2-SAT where a parameter is the number of unsatisfied clauses; on the other hand, when allowing the $n^{O(1)}$ factor to be super-linear, there exists an $O^*(2.32^k)$-time algorithm [23]. These two algorithms are not comparable: the former runs faster when the input is large but the latter runs faster when the parameter is large. Typically, only algorithms with the smallest $f(k)$ factor or the smallest $n^{O(1)}$ factor have been studied. However, if there were three algorithms running in time $O(8^k n)$, $O(4^k n^2)$, and $O(2^k n^3)$, all of them are incomparable: the first is fastest when $4^k < n$, the second is fastest when $2^k < n < 4^k$, and the third is fastest when $n < 2^k$. Do we need to develop algorithms with the smallest possible $f(k)$ factor for each n^d? We observe that *kernelization*, which is another basic research object of the parameterized complexity, is useful for avoiding this incomparability.

A kernelization algorithm (or *kernel*) for a parameterized problem is an algorithm that, given an instance (x, k) in time polynomial in $n = |x|$ and k, returns an equivalent instance (x', k') of the same problem such that $k' \leq k$ and $|x'| \leq g(k)$ for some function g. When the $n^{O(1)}$ factor in the running time is linear in n (i.e., $k^{O(1)}n$), it is called a *linear-time kernel*. If there is a kernel (and the problem is decidable), by solving the reduced instance exhaustively, we can obtain an FPT algorithm. Actually, the converse is also true; if there exists an $f(k)n^{O(1)}$-time FPT algorithm, there also exists a kernel of size $f(k)$. On the other hand, the existence of a polynomial-size (i.e., $|x'| \leq k^{O(1)}$) kernel is non-trivial and, actually, there are known to exist FPT problems which (unconditionally) do not have any sub-exponential-size kernels [4]. As in the case of FPT algorithms, the typical goal is to develop kernelization algorithms with a small size $g(k)$ (e.g., linear in k) and a fast running time (e.g., linear in n).

Compared with linear-time FPT algorithms, the number of studies for linear-time polynomial-size kernels is small. Examples include d-Hitting Set [29], Dominating Set on planar graphs [30, 16], $n-k$ Clique Covering [9], and Max Cut Above Edwards-Erdős Bound [14]. One of the major reasons is that when assuming the parameter k is a constant, which is often done when studying linear-time FPT algorithms, kernels become uninteresting because we cannot distinguish between $f(k)$ and $k^{O(1)}$. Nevertheless, improving the $n^{O(1)}$ factor in the running time of kernels is very important because such kernels can be used as preprocessing for FPT algorithms. Let us assume that we have a $k^{O(1)}n^d$-time polynomial-size kernel and an $f(k)n^{O(1)}$-time FPT algorithm. Then, by applying the FPT algorithm against the instance reduced by the kernelization, we obtain a $k^{O(1)}(f(k) + n^d)$-time FPT algorithm. Thus, the $n^{O(1)}$ factor of any FPT algorithm can be replaced by n^d. Therefore, if we have a linear-time polynomial-size kernel, we obtain a linear-time FPT algorithm that simultaneously achieves the smallest possible $f(k)$ factor (ignoring factors polynomial in k). This also implies that after obtaining a linear-time polynomial-size kernel, we can safely ignore the $n^{O(1)}$ factor and focus on improving the $f(k)$ factor only. Moreover, it can also be combined with another kernel of smaller size. Let us assume that we have a $k^{O(1)}n^d$-time polynomial-size kernel and a $g(k)$-size kernel. Then, by applying the second kernel against the instance reduced by the first kernelization, we obtain a $k^{O(1)}n^d$-time $g(k)$-size kernel. Therefore, in contrast to the case of FPT algorithms, we can always achieve the smallest size and the fastest running time simultaneously.

In this paper, we give a linear-time quadratic-size kernel for Feedback Vertex Set. This is the first linear-time polynomial-size kernel for this problem. Feedback Vertex Set is a problem to decide whether a given undirected graph has a vertex set of size at most a given parameter k whose removal makes the graph a forest. Feedback Vertex Set is one of the most comprehensively studied problems in the field of parameterized complexity and

many different FPT algorithms and kernels have been developed. Moreover, the problem was chosen as a target problem of the 1st Parameterized Algorithms and Computational Experiments (PACE) challenge[1]. Actually, this research is strongly motivated by the PACE challenge. The proposed methods are easy to implement, and a solver[2] based on the proposed methods won first place in the challenge.

The first FPT algorithm for Feedback Vertex Set was given by Downey and Fellows [13]. This algorithm and subsequent improved algorithms [21, 26] use the strategy to branch on short cycles and the $f(k)$ factor of the running time is not a single exponential in k. The first single-exponential FPT algorithms were obtained independently by Dehne et al. [11] and Guo et al. [15], and several improved algorithms have been obtained [8, 7, 22]. The current smallest $f(k)$ factor for deterministic algorithms is 3.62^k given by Kociumaka and Pilipczuk [22]. These single-exponential FPT algorithms use the *iterative compression* technique. For a graph with n vertices and m edges[3], a naive implementation of iterative compression requires n iterations and each iteration takes $f(k)\Omega(m)$ time. Therefore, the total running time is $f(k)\Omega(nm)$. For the case of Feedback Vertex Set, by combining it with 2-approximation algorithms [1, 3], we can solve the problem using only a single iteration; however, this increases the running time for one iteration to $f(2k)\Omega(m)$. Thus, for obtaining a linear-time FPT algorithm, the $f(k)$ factor needs to grow from 3.62^k to 3.62^{2k}. When allowing randomness, a simple $O(4^k km)$-time FPT algorithm using random sampling of edges was given by Becker et al. [2]. The current smallest $f(k)$ factor for randomized FPT algorithms is 3^k given by Cygan et al. [10]. This algorithm uses dynamic programming on tree-decompositions and takes $3^k k^{O(1)} n^2$ time after obtaining a tree-decomposition of width at most k. As discussed above, by using our linear-time polynomial-size kernel, we can obtain a $k^{O(1)}(3.62^k + m)$-time deterministic FPT algorithm and a $k^{O(1)}(3^k + m)$-time randomized FPT algorithm.

The first polynomial-size kernel was given by Burrage et al. [6]. The size of this kernel is $O(k^{11})$, which was improved to $O(k^3)$ by Bodlaender and Van Dijk [5], and to $O(k^2)$ by Thomassé [28]. Finally, Dell and Van Melkebeek [12] showed that there are no kernels of size $O(k^{2-\epsilon})$ for any constant $\epsilon > 0$ unless NP $\subseteq$ coNP/poly. The size of the current smallest kernel by Thomassé is $4k^2$ vertices and $8k^2$ edges. Although the precise running time of each of these kernels was not analyzed, we can easily check that all of them take at least $k^{O(1)}nm$ time. As discussed above, if there is a linear-time polynomial-size kernel, by combining it with the smallest kernel, we can achieve the linear running time and the smallest kernel size simultaneously. However, our linear-time quadratic-size kernel does not rely on such a combination.

Before providing a description of our kernel, we first give a brief description of a key idea behind the existing kernels. All the existing kernels for Feedback Vertex Set exploit *s-flowers*. A set of simple cycles is called an s-flower if each cycle contains the vertex s and no two cycles share any vertex other than s. If the degree of s is large and the graph is well-connected, there exists a large s-flower. Because the size of an s-flower (i.e., the number of cycles) gives a lower bound of the size of the minimum feedback vertex set that does not contain s, if it is larger than the parameter k, we can remove s and decrement k. Otherwise, the degree of s is small, or the graph is not well-connected. In the former case, we know that the graph is small, and in the latter case, we can apply another reduction rule.

[1] `https://pacechallenge.wordpress.com/`
[2] `https://github.com/wata-orz/fvs`
[3] If a graph has a feedback vertex set of size at most k, we have $m = O(kn)$.

In our kernel, instead of s-flowers, we exploit *k-submodular relaxation*, which is a recently developed technique for obtaining efficient FPT algorithms for various problems. The concept of k-submodular relaxation was introduced by Iwata, Wahlström, and Yoshida [19]. The k-submodular relaxation is a technique to obtain *half-integral* and *persistent* relaxations and many existing half-integral LP relaxations (e.g., the LP relaxation of Vertex Cover [24]) can be re-derived by this technique. If a problem admits such a relaxation, the branch-and-bound method gives an FPT algorithm. By applying k-submodular relaxation, they obtained an $O^*(4^k)$-time FPT algorithm for Feedback Vertex Set. A detailed description of the k-submodular for Feedback Vertex Set is given in Section 2. By exploiting k-submodular relaxation, we obtain a very simple kernel for Feedback Vertex Set, which is described in Section 3. The size of our kernel is $2k^2 + k$ vertices and $4k^2$ edges, which is smaller than the previous best of $4k^2$ vertices and $8k^2$ edges [28].

We observe that there is a strong relationship between the k-submodular relaxation of Feedback Vertex Set and s-flowers; the problem of computing a maximum s-flower is the integral dual of the k-submodular relaxation of Feedback Vertex Set. This resembles the situation for Almost 2-SAT. For Almost 2-SAT, Raman et al. [25] obtained an $O^*(9^k)$-time FPT algorithm by a reduction to Vertex Cover above Maximum Matching, and then both the $f(k)$ factor [23] and $n^{O(1)}$ factor [18, 27] were improved by a reduction to Vertex Cover above LP. Here, the maximum matching is the integral dual of the LP relaxation of Vertex Cover. Because the fractional minimum of the primal LP is always at least the integral maximum of the dual LP, by using the half-integral relaxation instead of the integral dual, we can obtain a better lower bound. Moreover, by using the half-integral relaxations, we can directly exploit the persistency of the relaxations.

For obtaining linear-time kernel, we give a max-flow-like augmenting-path algorithm for solving the k-submodular relaxation of Feedback Vertex Set in Section 4. This is the most technical part of the paper. Our algorithm can compute a minimum solution in $O(km)$ time. By combining this algorithm with the simple kernel, we give a linear-time kernel in Section 5. Due to space limitations, some of the proofs are omitted. Lemmas with omitted proofs are marked with ($\star$) and these proofs can be found in the full version [17].

We note that this algorithm can be used not only for the linear-time kernel but also for improving the $n^{O(1)}$ factor of the $O^*(4^k)$-time FPT branch-and-bound algorithm for Feedback Vertex Set. By applying k-submodular relaxations, $O^*(4^k)$-time FPT algorithms for two general versions, Subset Feedback Vertex Set and Group Feedback Vertex Set, have been obtained [19]. Following up our work, Iwata, Yamaguchi, and Yoshida [20] obtained linear-time FPT algorithms for many problems including these general versions of Feedback Vertex Set by developing efficient augmenting-path algorithm for solving k-submodular relaxations in general.

2 Preliminaries

2.1 Definitions

A *multiplicity function* of a multiset S is denoted by $\mathbf{1}_S$; e.g., when $S = \{a, a, b\}$, $\mathbf{1}_S(a) = 2$, $\mathbf{1}_S(b) = 1$, and $\mathbf{1}_S(c) = 0$. Let $f : U \to \mathbb{R}$ be a function. For a multiset S, we write the sum of $f(a)$ over $a \in S$ as $f(S) = \sum_{a \in S} f(a)$; e.g., when $S = \{a, a, b\}$, $f(S) = 2f(a) + f(b)$. We denote the preimage of $i \in \mathbb{R}$ under f by $f^{-1}(i) = \{a \in U \mid f(a) = i\}$.

Let $G = (V, E)$ be an undirected graph. We assume that G may contain a self-loop and multiple edges. We will often denote the number of vertices by n and the number of edges by m. We denote the set of edges incident to a vertex v by $\delta_G(v)$ and define the *degree*

of v as $d_G(v) = |\delta_G(v)|$. Here, we note that multiple edges contribute to the degree by its multiplicity, and we never refer to the degree of a vertex having a self-loop. We omit the subscript G if it is clear from the context. An edge $e \in E$ is called a *bridge* if its removal increases the number of connected components.

For a vertex set S, we denote the graph obtained by removing S and their incident edges by $G - S$. When S is a singleton $\{v\}$, we simply write $G - v$. A vertex set $S \subseteq V$ is called a *feedback vertex set* if $G - S$ is a forest. We denote the size of the minimum feedback vertex set of G by $\mathrm{fvs}(G)$.

A *walk* is an ordered list $(v_0, e_1, v_1, e_2, \ldots, v_{\ell-1}, e_\ell, v_\ell)$ such that $\ell \geq 1$ and each edge e_i connects vertices v_{i-1} and v_i. Note that it may contain a vertex or an edge multiple times. For a walk $W = (v_0, e_1, \ldots, v_\ell)$, we denote the multiset of vertices appearing on W by $V(W) = \{v_0, \ldots, v_\ell\}$ and the multiset of edges appearing on W by $E(W) = \{e_1, \ldots, e_\ell\}$.

2.2 Basic Reductions

We introduce basic reductions that have been commonly used in kernelization algorithms for Feedback Vertex Set [6, 5, 28]. The correctness of these reductions is almost trivial.

1. If there is a vertex v containing a self-loop, remove v and decrease k by one.
2. If there is a vertex of degree at most one, remove it.
3. If there is a vertex of degree two, remove it and connect its two neighbors by an edge.
4. If two vertices are connected by more than two edges, replace these edges with a double edge.

Note that rule 3 can remove a vertex that is only incident to a double edge; in this case, it creates a self-loop on its neighbor. These basic reductions never increase the degree of any vertex and can be fully applied in $O(m)$ time. After the reduction, the obtained graph has no self-loops and has minimum degree at least three.

We will use the following lemma to bound the size of the kernel. This is a general version of the lemma in [28], and a modified proof can be found in the full version [17].

▶ **Lemma 1** (Thomassé [28]). *If a graph without self-loops satisfies both of the following for an integer d, the size of the minimum feedback vertex set is larger than k:*

- *At least one of $n > dk + k$ or $m > 2dk$ holds; and*
- *for any $v \in V$, it holds that $3 \leq d(v) \leq d$.*

In [28], a kernel of $4k^2$ vertices and $8k^2$ edges is obtained by applying the lemma against $d = 4k - 1$. In the next section, we obtain a kernel of $2k^2 + k$ vertices and $4k^2$ edges by applying the lemma against $d = 2k$.

2.3 k-submodular Relaxation of Feedback Vertex Set

A walk $W = (v_0, e_1, \ldots, v_\ell)$ is called an *s-cycle* if $v_0 = v_\ell = s$, $v_i \neq s$ for all $i \in \{1, \ldots, \ell - 1\}$, $e_i \neq e_{i+1}$ for any $i \in \{1, \ldots, \ell - 1\}$ (i.e., there are no U-turns), and each edge is contained in the walk at most twice. For example, walks $(s, e_1, u, e_2, v, e_3, s)$ and $(s, e_1, u, e_2, v, e_3, w, e_4, u, e_1, s)$ are s-cycles but a walk $(s, e_1, u, e_2, v, e_2, u, e_1, s)$ is not. Note that in this definition, we distinguish each of multiple edges; e.g., if there is only a single edge e between s and v, a walk (s, e, v, e, s) is not an s-cycle; however, if there is a double edge $\{e_1, e_2\}$ between s and v, a walk (s, e_1, v, e_2, s) is an s-cycle.

For a graph $G = (V, E)$ without self-loops and a vertex $s \in V$, a function $x : V \to \mathbb{R}_{\geq 0}$ is called an *s-cycle cover* if it satisfies that (1) $x(s) = 0$ and (2) for any s-cycle C, $x(V(C)) \geq 1$.

Note that $V(C)$ is the multiset of vertices on C, and therefore if $x(v) = \frac{1}{2}$ holds for a vertex v contained twice in C, we have $x(V(C)) \geq 1$. The *size* of an s-cycle cover x is defined as $x(V)$, and when the size $x(V)$ is minimum among all the possible s-cycle covers, it is called a *minimum s-cycle cover*.

By introducing the idea of k-submodular relaxation, Iwata, Wahlström, and Yoshida [19] obtained the following lemma.

▶ **Lemma 2** (Iwata, Wahlström, and Yoshida [19]). *For any graph $G = (V, E)$ without self-loops and $s \in V$, the following holds:*

- *The size of any feedback vertex set of G that does not contain s is at least the size of the minimum s-cycle cover.*
- *There exists a minimum s-cycle cover that takes values $\{0, \frac{1}{2}, 1\}$ (half-integrality).*
- *If there exists a minimum feedback vertex set that does not contain s, then for any half-integral minimum s-cycle cover x, there also exists a minimum feedback vertex set S such that $x^{-1}(1) \subseteq S$ and $s \notin S$ (persistency).*

3 Simple Quadratic-size Kernel

In this section, we give a simple polynomial-time quadratic-size kernel for Feedback Vertex Set. By exploiting the persistency of the k-submodular relaxation, we give the following reduction rule called *s-cycle cover reduction*.

For a graph $G = (V, E)$, a vertex $s \in V$, and a half-integral minimum s-cycle cover x, create a graph $G' = (V, E')$ as follows. Let $X = x^{-1}(1)$ and let $B \subseteq \delta(s)$ be the set of bridges of $G - X$ connecting s and tree components of $G - X - s$. Then, G' is obtained from G by inserting a double edge between s and each of $v \in X$ and removing the edges B.

▶ **Lemma 3.** *For a graph $G = (V, E)$, a vertex $s \in V$, and a half-integral minimum s-cycle cover x, let $G' = (V, E')$ be a graph obtained by applying the s-cycle cover reduction. Then, $\mathrm{fvs}(G) = \mathrm{fvs}(G')$ holds.*

Proof. ($\geq$) Let S be a minimum feedback vertex set of G. Observe that all the inserted edges are between s and $X = x^{-1}(1)$. If $s \in S$, S is also a feedback vertex set of G'. Otherwise, from the persistency, we can assume that S contains all the vertices of X. Thus, S is also a feedback vertex set of G'.

($\leq$) Let S be a minimum feedback vertex set of G'. If $s \in S$, S is also a feedback vertex set of G. Otherwise, because all the vertices of X are connected to s by double edges in G', S must contain all of X. Because all the deleted edges are bridges in $G - X$, S is also a feedback vertex set of G. ◀

After applying this reduction, the degree of s can be bounded as the following shows.

▶ **Lemma 4.** *For a graph $G = (V, E)$, a vertex $s \in V$, and a half-integral minimum s-cycle cover x, let $G' = (V, E')$ be a graph obtained by applying the s-cycle cover reduction. Then, $d_{G'}(s) \leq 2x(V)$ holds.*

Proof. First, we show that x is also an s-cycle cover of G'. Let us assume that there is an s-cycle C of G' such that $x(C) < 1$. Because $x(v) = 1$ for $v \in X = x^{-1}(1)$, C contains none of X. Because all the inserted edges are incident to X, C is also an s-cycle of G, which is a contradiction.

For $i \in \{1, 2\}$, let N_i denote the set of vertices that are connected to s by edges of multiplicity i in G'. For each $v_i \in N_1$, we define a vertex w_i as follows.

Algorithm 1 Simple quadratic-size kernelization for Feedback Vertex Set.

```
1:  procedure Kernelize(G, k)
2:      while true do
3:          Apply the basic reductions
4:          if k < 0 then return NO
5:          if n ≤ 2k^2 + k and m ≤ 4k^2 then return (G, k)
6:          if ∀v ∈ V, d(v) ≤ 2k then return NO
7:          Pick a vertex s of degree larger than 2k
8:          Compute a half-integral minimum s-cycle cover x
9:          if x(V) > k then G ← G − s; k ← k − 1
10:         else apply the s-cycle cover reduction
```

If the edge sv_i is a bridge in $G' - X$, let C_i be the connected component of $G' - X - s$ containing v_i. Because the reduction removes all the bridges between s and tree components, C_i is not a tree. Thus, there exists an s-cycle contained in $C_i \cup \{s\}$ and, therefore, there must exist a vertex $w_i \in C_i$ with $x(w_i) = \frac{1}{2}$.

If sv_i is not a bridge in $G' - X$, there exists a path P_i from v_i to $N_1 \setminus \{v_i\}$ in $G' - X - s$. Fix an arbitrary path P_i and let w_i be the first vertex on the path such that $x(w_i) = \frac{1}{2}$. Because x is an s-cycle cover, there always exists such a vertex.

If $w_i = w_j$ holds for some $i \neq j$, there exists an s-cycle C such that $x(C) = \frac{1}{2}$, which is a contradiction. Therefore, all w_i are distinct. Thus, we have $d_{G'}(s) = |N_1| + 2|N_2| \leq |x^{-1}(\frac{1}{2})| + 2|x^{-1}(1)| = 2x(V)$. ◀

Now, we describe our simple quadratic-size kernelization algorithm (see Algorithm 1). First, we apply the basic reductions. If k becomes negative, we return a NO instance. If the graph becomes small enough, we return it. If all the vertices have degree at most $2k$, we return a NO instance. Otherwise, pick an arbitrary vertex s of degree larger than $2k$, and compute a half-integral minimum s-cycle cover x. If the size of the s-cycle cover is larger than k, we remove s and decrement k. Otherwise, we apply the s-cycle cover reduction.

▶ **Lemma 5.** *Algorithm 1 runs in $(k+m)^{O(1)}$ time and correctly computes (G', k') satisfying $k' \leq k$ and $\mathrm{fvs}(G) \leq k \Leftrightarrow \mathrm{fvs}(G') \leq k'$. The size of G' is at most $2k^2 + k$ vertices and $4k^2$ edges.*

Proof. It obviously holds that $k' \leq k$ and the size of G' is at most $2k^2 + k$ vertices and $4k^2$ edges. From Lemma 4, after applying the s-cycle cover reduction, the degree of s changes from more than $2k$ to at most $2k$. Therefore, the number of edges strictly decreases for each iteration. Thus, it stops in at most m iterations. Because each iteration can be done in time polynomial in k and m, the total running time is also polynomial in k and m.

Next, we show the correctness. By applying Lemma 1 against $d = 2k$, when the maximum degree is at most $2k$ and at least one of $n > 2k^2 + k$ and $m > 4k^2$ holds, $\mathrm{fvs}(G) > k$ holds. Thus, we can safely return a NO instance (line 6). From Lemma 2, if $x(V) > k$, there is no feedback vertex set of size at most k that does not contain s. Thus, we can safely remove s (line 9). The correctness of the s-cycle cover reduction follows from Lemma 3. ◀

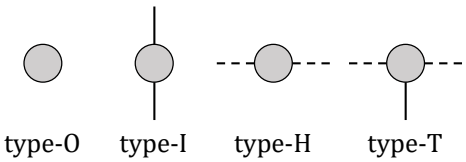


Figure 1 Four types of vertices.

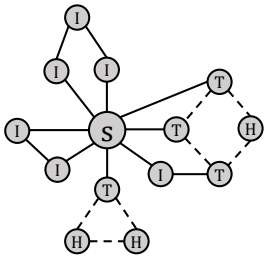


Figure 2 Example of the basic s-cycle packing.

4 Efficient Computation of a Half-integral Minimum s-cycle Cover

In this section, we prove the following theorem.

▶ **Theorem 6.** *Given a graph $G = (V, E)$ without self-loops, a vertex $s \in V$, and an integer k, in $O(km)$ time, we can compute a half-integral minimum s-cycle cover or conclude that there are no s-cycle covers of size at most $\frac{k}{2}$.*

First, we give several definitions. Let $\mathcal{C}_s$ denote the set of all s-cycles. A function $y : \mathcal{C}_s \to \mathbb{R}$ is called an *s-cycle packing* if for any vertex $v \in V \setminus \{s\}$, it holds that $\sum_{C \in \mathcal{C}_s} \mathbf{1}_{V(C)}(v) y(C) \leq 1$. The *size* of an s-cycle packing y is defined as $y(\mathcal{C}_s)$, and when the size $y(\mathcal{C}_s)$ is the maximum among all the possible s-cycle packings, it is called a *maximum s-cycle packing*. Because the problem of finding a maximum s-cycle packing is the LP dual of the problem of finding a minimum s-cycle cover, the size of the minimum s-cycle cover is equal to the size of the maximum s-cycle packing. Thus, if we can find a pair of an s-cycle cover x and an s-cycle packing y of the same size, we can confirm that x is a minimum s-cycle cover and y is a maximum s-cycle packing.

▶ **Definition 7.** A function $f : E \to \{0, \frac{1}{2}, 1\}$ is called a *basic s-cycle packing* if it satisfies the following three conditions.

1. For any $e \in \delta(s)$, $f(e) \in \{0, 1\}$.
2. Each vertex $v \in V \setminus \{s\}$ satisfies exactly one of the following four conditions (see Figure 1):
   a. $f(e) = 0$ for all edges $e \in \delta(v)$ (called type-O);
   b. $f(e) = 1$ for exactly two edges $e \in \delta(v)$ and $f(e) = \frac{1}{2}$ for none of $e \in \delta(v)$ (called type-I);
   c. $f(e) = 1$ for none of $e \in \delta(v)$ and $f(e) = \frac{1}{2}$ for exactly two edges $e \in \delta(v)$ (called type-H);
   d. $f(e) = 1$ for exactly one edge $e \in \delta(v)$ and $f(e) = \frac{1}{2}$ for exactly two edges $e \in \delta(v)$ (called type-T).
3. For each vertex $v \in V \setminus \{s\}$ of type-H or type-T, the cycle obtained by following edges of value $\frac{1}{2}$ from v contains an odd number of type-T vertices.

We call the cycle in the third condition the *half-integral cycle of v*. The size of a basic s-cycle packing f is defined as $\frac{1}{2} f(\delta(s))$. Figure 2 illustrates an example of the basic s-cycle packing, where solid lines denote edges of value 1, and dotted lines denote edges of value $\frac{1}{2}$.

▶ **Lemma 8** ($\star$). *If there exists a basic s-cycle packing of size k, there also exists an s-cycle packing of size k.*

Note that this lemma only says that the size of the maximum basic s-cycle packing is always at most the size of the maximum s-cycle packing and does not imply these two are equal. The equality is shown at the end of this section.

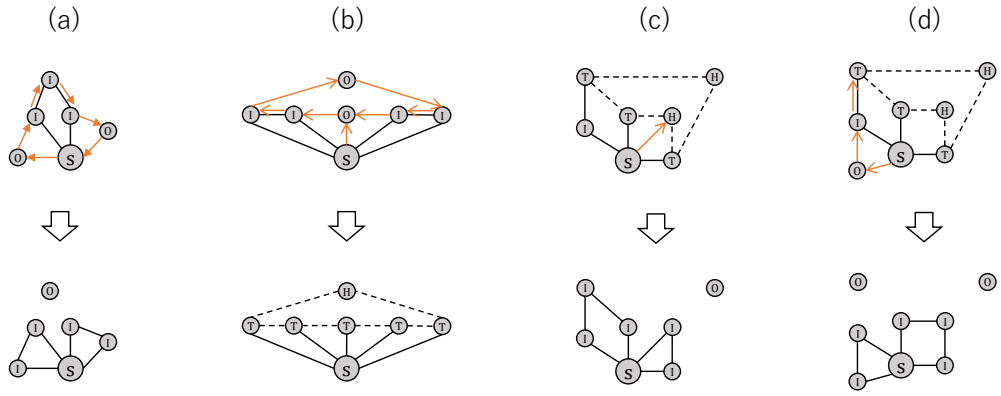


Figure 3 Examples of augmenting walks.

▶ **Definition 9.** For a basic s-cycle packing f, a walk $W = (v_0, e_1, \ldots, v_\ell)$ is called an *f-augmenting walk* if it satisfies all the following conditions.

1. We have $v_0 = s$.
2. We have $f(e_1) = 0$.
3. All the edges $\{e_1, \ldots, e_\ell\}$ are distinct.
4. The vertices $\{v_0, \ldots, v_{\ell-1}\}$ are distinct (the last vertex v_ℓ can be identical to v_i for some $i < \ell$).
5. For each $i \in \{1, \ldots, \ell - 1\}$, exactly one of the following holds:
   a. v_i is type-O;
   b. v_i is type-I and $f(e) = 1$ holds for at least one of $e \in \{e_i, e_{i+1}\}$.
6. If $v_\ell = v_i$ for some $i < \ell$, exactly one of the following holds:
   a. $v_\ell = s$ and $f(e_\ell) = 0$;
   b. v_ℓ is type-O;
   c. v_ℓ is type-I and $f(e) = 1$ holds for at least one of $e \in \{e_i, e_\ell\}$.
7. If $v_\ell \notin \{v_0, \ldots, v_{\ell-1}\}$, v_ℓ is type-H or type-T.

For a basic s-cycle packing f and an f-augmenting walk $W = (v_0, e_1, \ldots, v_\ell)$, let $f_W : E \to \{0, \frac{1}{2}, 1\}$ be a function defined as follows. First, set $f_W(e) = f(e)$ for all edges $e \in E$. If $v_\ell \neq s$ and $v_\ell = v_i$ holds for some $i < \ell$, let $h = i$; otherwise, let $h = \ell$. Then, for each edge $e \in \{e_1, \ldots, e_h\}$, set $f_W(e) = 1 - f(e)$. If $v_\ell = s$, we finish (see Figure 3-(a)). Otherwise, we further modify f_W depending on the type of v_ℓ.

(Case 1) If v_ℓ is type-O or type-I, for each edge $e \in \{e_{h+1}, \ldots, e_\ell\}$, set $f_W(e) = \frac{1}{2}$ (see Figure 3-(b)).

(Case 2) If v_ℓ is type-H, let C be the half-integral cycle of v_ℓ and let $\{t_0 = v_\ell, t_1, \ldots, t_{q-1}\}$ be the vertex set consisting of the vertex v_ℓ and the type-T vertices on C ordered along C (i.e., v_ℓ is located on the path from t_{q-1} to t_1 along the cycle C). We use the notation $t_q = t_0$. Let P_i be the path from t_i to t_{i+1} along the cycle C. For each even i, set $f_W(e) = 0$ for all the edges on P_i, and for each odd i, set $f_W(e) = 1$ for all the edges on P_i (see Figure 3-(c)).

(Case 3) If v_ℓ is type-T, let C be the half-integral cycle of v_ℓ and let $\{t_0 = v_\ell, t_1, \ldots, t_{q-1}\}$ be the type-T vertices on C ordered along C. Then, we proceed in exactly the same way as in case 2 (see Figure 3-(d)). We note that, in case 2, q is even; thus, v_ℓ is connected to t_{q-1} by edges of value one in f_W. On the other hand, in case 3, q is odd; thus, v_ℓ is connected to none of t_1 and t_{q-1}.

We call this operation that creates f_W from f as *augmenting f along W*.

Algorithm 2 Algorithm for computing an f-augmenting walk.

```
1: procedure FindAugmentingWalk(G, s, f)
2:     S ← {s}
3:     prev(v) = ϵ for all v ∈ V
4:     while S ≠ ∅ do
5:         Pick a vertex u ∈ S and remove u from S
6:         for e = uv ∈ δ(u) do
7:             if e = prev(u) then continue
8:             if e ∈ δ(s) and f(e) = 1 then continue
9:             if u is type-I and f(prev(u)) = f(e) = 0 then continue
10:            if v is type-H or type-T then
11:                prev(v) ← e
12:                return the walk from s to v along the search tree
13:            else if prev(v) = ϵ then
14:                prev(v) ← e; S ← S ∪ {v}
15:            else if v is type-O or f(prev(v)) + f(e) ≥ 1 then
16:                return the walk s → u → v → w along the search tree
17:    return NO
```

▶ **Lemma 10 (⋆).** *For a basic s-cycle packing f and an f-augmenting walk $W = (v_0, e_1, \ldots, v_\ell)$, let $f_W : E \to \{0, \frac{1}{2}, 1\}$ be the function obtained by augmenting f along W. Then, f_W is a basic s-cycle packing. Moreover, if $v_\ell = s$, the size of f_W is the size of f plus one; and otherwise, the size of f_W is the size of f plus $\frac{1}{2}$.*

Now, we give an algorithm to compute an f-augmenting walk (see Algorithm 2). First, we initialize a set S and a table $\mathrm{prev} : V \to E \cup \{\epsilon\}$. The set S stores vertices we need to process and is initialized to $\{s\}$. We ensure that only the vertex s and vertices of type-O or type-I are stored in S. The table $\mathrm{prev}(v)$ represents an edge to the parent of v in the search tree and initialized to the dummy edge ϵ, which indicates that the vertex is not visited (or the vertex is the root s). Then, while S is not empty, pick up an arbitrary vertex u from S and process each incident edge $e = uv \in \delta(u)$ as described below. If S becomes empty, the algorithm returns NO.

First, we check whether the edge $e = uv$ is valid by testing the following three conditions. If $e = \mathrm{prev}(u)$, because we have already processed this edge, we skip it. If e is incident to s and $f(e) = 1$, because such an edge cannot be used in an augmenting walk, we skip it. Note that, when $v = s$, at least one of these two conditions are satisfied. Similarly, if u is type-I and both of $f(\mathrm{prev}(u))$ and $f(e)$ are zero, because we cannot use both of $\mathrm{prev}(u)$ and e simultaneously, we skip it.

If v is type-H, or type-T, we return the walk from s to v in the search tree by using the table prev. If $\mathrm{prev}(v) = \epsilon$, we set $\mathrm{prev}(v) = e$ and insert it to S. If v is already visited and v is type-O, let w be the lowest common ancestor of u and v in the search tree. Then, we return the walk obtained by going down from s to u along the search tree, jumping from u to v by the edge e, and then by going up from v to w along the search tree. If v is already visited and v is type-I, we basically do the same; however, we need one additional constraint. If both of $f(\mathrm{prev}(v))$ and $f(e)$ are zero, the walk created as above does not satisfy the condition 6(c) of Definition 9; thus, we skip the edge without returning the walk.

From the construction of our algorithm, we obtain the following lemma.

▶ **Lemma 11.** *A walk returned by Algorithm 2 is an f-augmenting walk.*

Note that this lemma does not say that Algorithm 2 returns an f-augmenting walk whenever there exists an f-augmenting walk; it only says that if the algorithm returns a walk, it is an f-augmenting walk, and the algorithm might return NO even when there exists an f-augmenting walk. We now show that, if the algorithm returns NO, we can construct an s-cycle cover x whose size is equal to the size of f. From Lemma 8 and the LP duality of s-cycle packings and s-cycle covers, the size of a basic s-cycle packing is always at most the size of an s-cycle cover. Therefore, this equality implies that the current basic s-cycle packing f is the maximum and the constructed s-cycle packing x is the minimum. This also implies that when the algorithm returns NO, there are no f-augmenting walks.

To construct such an s-cycle cover x, we first prove a property of the table prev. We call a vertex $v \in V$ *reachable* if $v = s$ or $\mathrm{prev}(v) \neq \epsilon$. For each edge $e \in \delta(s)$ with $f(e) = 1$, by following edges of value 1 from e, we can obtain a simple cycle returning to s or a simple path to a type-T vertex. We denote such a cycle or a path by W_e. Note that when W_e is a cycle, $W_e = W_{e'}$ for another edge $e' \in \delta(s)$.

▶ **Lemma 12 (⋆).** *If Algorithm 2 returns NO, exactly one of the following holds for each edge $e \in \delta(s)$ with $f(e) = 1$:*

1. $\mathrm{prev}(v) = \epsilon$ *for any vertex* $v \in V(W_e)$*;*
2. W_e *is a cycle, all the vertices on* W_e *are reachable, and exactly one vertex* $v \in V(W_e) \setminus \{s\}$ *satisfies* $\mathrm{prev}(v) \notin E(W_e)$*.*

When Algorithm 2 returns NO, by using the obtained table prev, we construct a function $x : V \to \{0, \frac{1}{2}, 1\}$ as follows. For each edge $e = su \in \delta(s)$ with $f(e) = 1$, if W_e is a cycle satisfying the second condition of Lemma 12, we set $x(v) = 1$ for the unique vertex v satisfying $\mathrm{prev}(v) \notin E(W_e)$. Otherwise, we set $x(u) = \frac{1}{2}$. If $x(u)$ is already set to $\frac{1}{2}$, e.g., $W_{e_1} = W_{e_2} = (s, e_1, u, e_2, s)$ for a double edge $\{e_1, e_2\}$, we set $x(u) = 1$.

▶ **Lemma 13 (⋆).** *If Algorithm 2 returns NO, the function x is a minimum s-cycle cover.*

Proof of Theorem 6. Because each augmentation increases the size of f by at least $\frac{1}{2}$, after $k+1$ steps, we can obtain a half-integral minimum s-cycle cover of size at most $\frac{k}{2}$, or conclude that there are no s-cycle covers of size at most $\frac{k}{2}$. Because Algorithm 2 runs in $O(m)$ time, the total running time is $O(km)$. ◀

5 Linear-time Quadratic-size Kernel

In this section, we improve the running time of the quadratic-size kernel presented in Section 3 to $O(k^4 m)$. By using the $O(km)$-time algorithm for computing the minimum s-cycle cover presented in Section 4, each iteration can be done in $O(km)$ time. However, because the number of iterations is only bounded by $O(m)$, the total running time becomes $O(km^2)$. We show that, by a slight modification to Algorithm 1, the number of iteration can be bounded by $O(k^3)$; thus, the total running time becomes $O(k^4 m)$.

We add the following two rules just after line 6 of Algorithm 1. The safeness of these two rules is almost trivial.

- If there is a vertex v incident to more than k double edges, remove v, decrement k, and continue the iteration.
- If there are more than k^2 double edges, return NO.

For bounding the number of iterations, we use the following lemma.

▶ **Lemma 14.** *For a graph $G = (V, E)$ of minimum degree at least three, a vertex $s \in V$, and a half-integral minimum s-cycle cover x, if $2x(V) < d_G(s)$ holds, then $x^{-1}(1) \neq \emptyset$.*

Proof. From Lemma 4, for a graph G' obtained by applying the s-cycle cover reduction, it holds that $d_{G'}(s) \leq 2x(V)$. When $x^{-1}(1) = \emptyset$, the reduction inserts no new edges and only removes the bridges of G connecting s and tree components of $G - s$. Because the graph $G - s$ has minimum degree at least two, it has no tree components. Thus, we have $G' = G$, which is a contradiction. ◀

Now, we can prove the upper bound on the number of iterations.

▶ **Lemma 15.** *The modified Algorithm 1 stops in $O(k^3)$ iterations.*

Proof. Initially, all the double edges are blue, and after applying the s-cycle cover reduction, we color all the double edges incident to s red (not only newly inserted double edges but also blue colored edges are recolored to red). The other double edges, which are created by the deletion of degree two vertices in the basic reductions, are colored blue. Let α denote the number of red double edges and β denote the number of vertices of degree larger than $2k$ and incident to at least one red double edge. Let k_0 be the initial value of k and ϕ be a potential defined as $\phi = (2k_0^2 + 4k_0 + 3)k - 2\alpha + \beta$. Observe that, because red double edges are created only by the s-cycle cover reduction, each vertex can be incident to at most $k_0 + 1$ red double edges, and that the number of red double edges is always at most $k_0^2 + k_0$ (at most k_0^2 edges before applying the s-cycle cover reduction and the reduction can create at most k_0 red double edges).

Initially, there are no red edges; thus, the initial potential is $(2k_0^2 + 4k_0 + 3)k_0 = O(k_0^3)$. If ϕ becomes negative, we have $k < 0$ or $\alpha > k_0^2 k \geq k^2$. Thus, the algorithm returns NO.

When k is decremented, α can decrease by at most $k_0 + 1$. Because there are at most $k_0^2 + k_0$ red double edges, β can increase by at most $2k_0^2 + 2k_0$. Thus, ϕ decreases by at least $(2k_0^2 + 4k_0 + 3) - 2(k_0 + 1) - (2k_0^2 + 2k_0) \geq 1$.

When applying the s-cycle cover reduction, if the reduction creates c (≥ 1) new red double edges, α increases by c and β can increase by at most c. Thus, ϕ decreases by at least $2c - c = c \geq 1$. After applying the s-cycle cover reduction, from Lemma 14, s is incident to at least one double edge. Thus, if the reduction does not create any new red double edges, s must be incident to at least one red double edge before the reduction. From 4, the degree of s becomes at most $2k$ after the reduction. Therefore β decreases by at least one; thus, ϕ decreases by at least one.

Now, we have shown that each iteration decreases the potential ϕ by at least one. Because ϕ is initially $O(k^3)$ and is always non-negative, the number of iterations is $O(k^3)$. ◀

References

1. Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discrete Math.*, 12(3):289–297, 1999.
2. Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *J. Artif. Intell. Res.*, 12:219–234, 2000.
3. Ann Becker and Dan Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artif. Intell.*, 83(1):167–188, 1996.
4. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

5 Hans L. Bodlaender and Thomas C. van Dijk. A cubic kernel for feedback vertex set and loop cutset. *Theory Comput. Syst.*, 46(3):566–597, 2010.

6 Kevin Burrage, Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, Shev Mac, and Frances A. Rosamond. The undirected feedback vertex set problem has a poly(k) kernel. In *IWPEC 2006*, pages 192–202, 2006.

7 Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015.

8 Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008.

9 Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *WG 2004*, pages 257–269, 2004.

10 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS 2011*, pages 150–159, 2011.

11 Frank K. H. A. Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.*, 41(3):479–492, 2007.

12 Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014.

13 Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.

14 Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. In *ISAAC 2016*, pages 31:1–31:13, 2016.

15 Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

16 Torben Hagerup. Simpler linear-time kernelization for planar dominating set. In *IPEC 2011*, pages 181–193, 2011.

17 Yoichi Iwata. Linear-time kernelization for feedback vertex set. *CoRR*, abs/1608.01463, 2016. Full version of this paper. URL: `https://arxiv.org/abs/1608.01463`.

18 Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time FPT algorithms via network flow. In *SODA 2014*, pages 1749–1761, 2014.

19 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016.

20 Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. Linear-time FPT algorithms via half-integral non-returning A-path packing. *CoRR*, abs/1704.02700, 2017.

21 Iyad A. Kanj, Michael J. Pelsmajer, and Marcus Schaefer. Parameterized algorithms for feedback vertex set. In *IWPEC 2004*, pages 235–247, 2004.

22 Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Inf. Process. Lett.*, 114(10):556–560, 2014.

23 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014.

24 G. Nemhauser and L. Trotter. Vertex Packing: Structural Properties and Algorithms. *Math. Program.*, 8:232–248, 1975.

25 Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Paths, flowers and vertex cover. In *ESA 2011*, pages 382–393, 2011.

26 Venkatesh Raman, Saket Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Trans. Algorithms*, 2(3):403–415, 2006.

27 M. S. Ramanujan and Saket Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In *SODA 2014*, pages 1739–1748, 2014.

28 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2), 2010.

29 René van Bevern. Towards optimal and expressive kernelization for d-hitting set. *Algorithmica*, 70(1):129–147, 2014.

30 René van Bevern, Sepp Hartung, Frank Kammer, Rolf Niedermeier, and Mathias Weller. Linear-time computation of a linear problem kernel for dominating set on planar graphs. In *IPEC 2011*, pages 194–206, 2011.

Exact Algorithms via Multivariate Subroutines*

Serge Gaspers[†1] **and Edward J. Lee**[2]

2 **The University of New South Wales, Sydney, Australia; and Data61, CSIRO, Sydney, Australia**
sergeg@cse.unsw.edu.au

2 **The University of New South Wales, Sydney, Australia; and Data61, CSIRO, Sydney, Australia**
e.lee@unsw.edu.au

Abstract

We consider the family of Φ-Subset problems, where the input consists of an instance I of size N over a universe U_I of size n and the task is to check whether the universe contains a subset with property Φ (e.g., Φ could be the property of being a feedback vertex set for the input graph of size at most k). Our main tool is a simple randomized algorithm which solves Φ-Subset in time $(1+b-\frac{1}{c})^n N^{O(1)}$, provided that there is an algorithm for the Φ-Extension problem with running time $b^{n-|X|}c^k N^{O(1)}$. Here, the input for Φ-Extension is an instance I of size N over a universe U_I of size n, a subset $X \subseteq U_I$, and an integer k, and the task is to check whether there is a set Y with $X \subseteq Y \subseteq U_I$ and $|Y \setminus X| \leq k$ with property Φ. We also derandomize this algorithm at the cost of increasing the running time by a subexponential factor in n, and we adapt it to the enumeration setting where we need to enumerate all subsets of the universe with property Φ. This generalizes the results of Fomin et al. [STOC 2016] who proved them for the case $b = 1$. As case studies, we use these results to design faster deterministic algorithms for

- checking whether a graph has a feedback vertex set of size at most k,
- enumerating all minimal feedback vertex sets,
- enumerating all minimal vertex covers of size at most k, and
- enumerating all minimal 3-hitting sets.

We obtain these results by deriving new $b^{n-|X|}c^k N^{O(1)}$-time algorithms for the corresponding Φ-Extension problems (or the enumeration variant). In some cases, this is done by simply adapting the analysis of an existing algorithm, in other cases it is done by designing a new algorithm. Our analyses are based on Measure and Conquer, but the value to minimize, $1+b-\frac{1}{c}$, is unconventional and leads to non-convex optimization problems in the analysis.

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

Keywords and phrases enumeration algorithms, exponential time algorithms, feedback vertex set, hitting set

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.69

1 Introduction

In exponential-time algorithmics [8], the aim is to design algorithms for NP-hard problems with the natural objective to minimize their running times. In this paper, we consider a

* For a full version of the paper see http://arxiv.org/abs/1704.07982 [11].

† Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 69; pp. 69:1–69:13

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

broad class of subset problems, where for an input instance I on a universe U_I, the question is whether there is a subset S of the universe satisfying certain properties. For example, in the Feedback Vertex Set problem, the input instance consists of a graph $G = (V, E)$ and an integer k, the universe is the vertex set and the property to be satisfied by a subset S is the conjunction of "$|S| \leq k$" and "$G - S$ is acyclic".

More formally, and using definitions from [5], an *implicit set system* is a function Φ that takes as input a string $I \in \{0,1\}^*$ and outputs a set system $(U_I, \mathcal{F}_I)$, where U_I is a universe and $\mathcal{F}_I$ is a collection of subsets of U_I. The string I is referred to as an *instance* and we denote by $|U_I| = n$ the size of the universe and by $|I| = N$ the size of the instance. We assume that $N \geq n$. The implicit set system Φ is *polynomial time computable* if (a) there exists a polynomial time algorithm that given I produces U_I, and (b) there exists a polynomial time algorithm that given I, U_I and a subset S of U_I determines whether $S \in \mathcal{F}_I$. All implicit set systems discussed in this paper are polynomial time computable.

Φ-Subset	
Input:	An instance I
Output:	A set $S \in \mathcal{F}_I$ if one exists.

Φ-Extension	
Input:	An instance I, a set $X \subseteq U_I$, and an integer k.
Question:	Does there exists a subset $S \subseteq (U_I \setminus X)$ such that $S \cup X \in \mathcal{F}_I$ and $\lvert S\rvert \leq k$?

In recent work, Fomin et al. [5] showed that $c^k N^{O(1)}$ time algorithms ($c \in O(1)$) for Φ-Extension lead to competitive exponential-time algorithms for many Φ-Subset problems. The main tool was a simple randomized algorithm which solves Φ-Subset in time $(2 - \frac{1}{c})^n N^{O(1)}$ if there is an algorithm that solves Φ-Extension in time $c^k N^{O(1)}$. A derandomization was also given, turning the randomized algorithm into a deterministic one at the cost of a $2^{o(n)}$ factor in the running time. The method was also adapted to enumeration algorithms and combinatorial upper bounds. This framework, together with a large body of work in parameterized algorithmics [3], where $c^k N^{O(1)}$ time algorithms are readily available for many subset problems, led to faster algorithms for around 30 decision and enumeration problems.

In this paper, we extend the results of Fomin et al. [5] and show that a $b^{n-|X|}c^k N^{O(1)}$ time algorithms ($b, c \in O(1)$) for Φ-Extension lead to randomized $(1 + b - \frac{1}{c})^n N^{O(1)}$ time algorithms for Φ-Subset. Our result can be similarly derandomized and adapted to the enumeration setting. Observe that for $b = 1$, the results of [5] coincide with ours, but that ours have the potential to be more broadly applicable and to lead to faster running times. The main point is that if we use a $c^k N^{O(1)}$ time algorithm as a subroutine to design an algorithm exponential in n, we might as well allow a small exponential factor in n in the running time of the subroutine.

Similar as in [5], the Φ-Extension problem can often be solved by preprocessing the elements in X in a simple way and then using an algorithm for a subset problem. In the case of Feedback Vertex Set, the vertices in X can simply be deleted from the input graph. Whereas the literature is rich with $c^k N^{O(1)}$ time algorithms for subset problems, algorithms with running times of the form $b^n c^k N^{O(1)}$ with $b > 1$ are much less common.[1] One issue is

[1] One notable exception is by Eppstein [4], who showed that all maximal independent sets of size at most k in a graph on n vertices can be enumerated in time $(4/3)^n (81/64)^k n^{O(1)}$.

that there is, in general, no obviously best trade-off between the values of b and c for such algorithms. However, the present framework gives us a precise objective: we should aim for values of b and c that minimize the base of the exponent, $(1 + b - \frac{1}{c})$.

Our applications consist of three case studies centered around some of the most fundamental problems considered in [5], feedback vertex sets and hitting sets. For the first case study, we considered the Feedback Vertex Set problem: given a graph G and an integer k, does G have a feedback vertex set of size at most k? For this problem, we re-analyze the running time of the algorithm from [6]. In [6, 10], the algorithm was analyzed using Measure and Conquer: using a measure that is upper bounded by αn and aiming for a running time of $2^{\alpha n} n^{O(1)}$ the analysis of the branching cases led to constraints lower bounding the measure and the objective was to minimize α subject to these constraints. In our new analysis, we add an additive term $w_k \cdot k$ to the measure and adapt the constraints accordingly. If all constraints are satisfied, we obtain a running time of $2^{\alpha n + w_k k} n^{O(1)}$. Our framework naturally leads us to minimize $2^{\alpha} - 2^{-w_k}$. This approach leads to a $O(1.5422^n \cdot 1.2041^k)$ time algorithm, which, combined with our framework gives a deterministic $O(1.7117^n)$ time algorithm for Feedback Vertex Set. This improves on previous results giving $O(1.8899^n)$ [13], $O(1.7548^n)$ [6], $O(1.7356^n)$ [14], $O(1.7347^n)$ [9], and $O(1.7216^n)$ [5] time algorithms for the problem. We note that adapting the analysis of other existing exact and parameterized algorithms did not give faster running times. Also, if we allow randomization, the $O(1.6667^n)$ time algorithm by [5] (which can also be achieved using our framework) remains fastest.

Our second case study is more involved. Simply using an existing algorithm and adapting the measure was not sufficient to improve upon the best known enumeration algorithms (and combinatorial upper bounds) for minimal feedback vertex sets. Here, the task is, given a graph G, to output each feedback vertex set that is not contained in any other feedback vertex set. We design a new algorithm for enumerating all minimal feedback vertex sets. We also need a new combinatorial upper bound for the number of minimal vertex covers of size at most k to handle one special case in the enumeration of minimal feedback vertex sets.[2] We obtain a $O(1.7183^n \cdot 1.1552^k)$ time algorithm for enumerating all minimal feedback vertex sets. Our framework thus leads to a running time of $O(1.8527^n)$, improving on the previous best bound of $O(1.8638^n)$ [6]. The current best lower bound for the number of minimal feedback vertex sets is $O(1.5926^n)$ [6]. We would like to highlight that the enumeration of minimal feedback vertex sets is completely out of scope for the more restricted framework of [5]: the number of minimal feedback vertex sets of size at most k cannot be upper bounded by $c^k n^{O(1)}$ for any $c \in O(1)$, as evidenced by a disjoint union of k cycles of length n/k.

Our last case study gives a new algorithm for enumerating all minimal 3-hitting sets, also known as minimal transversals of rank-3 hypergraphs. These are minimal sets S of vertices of a hypergraph where each hyperedge has size at most 3 such that every hyperedge contains at least one vertex of S. We re-analyze an existing algorithm [2] for this enumeration problem, adapting the measure in a similar way as in the first case study, and we obtain a multivariate running time of $O(1.5135^n \cdot 1.1754^k)$, leading to an $O(1.6627^n)$ time enumeration algorithm. This breaks the natural time bound of $O(1.6667^n)$ of the previously fastest algorithm [5]. The current best lower bound gives an infinite family of rank-3 hypergraphs with $\Omega(1.5848^n)$ minimal transversals [2].

Lastly, all random selection is done from a uniform distribution and all randomized algorithms in this paper are Monte Carlo algorithms with one-sided error. On No-instances

[2] Previous work [1, 4] focused on small maximal independent sets, whose bounds were insufficient for us. We need better bounds on large maximal independent sets or small minimal vertex covers.

they always return No, and on Yes-instances they return Yes (or output a certificate) with probability $> \frac{1}{2}$.

2 Results

Our first main result gives exponential-time randomized algorithms for Φ-Subset based on single-exponential multivariate algorithms for Φ-Extension with parameter k.

▶ **Theorem 1.** *If there is an algorithm for* Φ-Extension *with running time* $b^{n-|X|}c^k N^{O(1)}$ *then there is a randomized algorithm for* Φ-Subset *with running time* $(1+b-\frac{1}{c})^n N^{O(1)}$.

The next main result derandomizes the algorithm of Theorem 1 at a cost of a subexponential factor in n in the running time.

▶ **Theorem 2.** *If there is an algorithm for* Φ-Extension *with running time* $b^{n-|X|}c^k N^{O(1)}$ *then there is an algorithm for* Φ-Subset *with running time* $(1+b-\frac{1}{c})^{n+o(n)} N^{O(1)}$.

We require the following notion of *(b,c)-uniform* to describe our enumeration algorithms. Let $c, b \geq 1$ be real valued constants and Φ be an implicit set system. Then Φ is *(b,c)-*uniform if for every instance I, set $X \subseteq U_I$, and integer $k \leq n - |X|$, the cardinality of the collection $\mathcal{F}^k_{I,X} = \{S \subseteq U_I \backslash X : |S| = k \text{ and } S \cup X \in \mathcal{F}_I\}$ is at most $b^{n-|X|}c^k n^{O(1)}$. Then the following theorem provides new combinatorial bounds for collections generated by (b, c)-uniform implicit set systems.

▶ **Theorem 3.** *Let* $c, b \geq 1$ *and* Φ *be an implicit set system. If* Φ *is* (b, c)*-uniform then* $|\mathcal{F}_I| \leq \left(1+b-\frac{1}{c}\right)^n n^{O(1)}$ *for every instance* I.

We say that an implicit set system is *efficiently* (b, c)*-uniform* if there exists an algorithm that given I, X and k enumerates all elements of $\mathcal{F}^k_{X,I}$ in time $b^{n-|X|}c^k N^{O(1)}$. In this case, we enumerate $\mathcal{F}_I$ in the same time, up to a subexponential factor in n.

▶ **Theorem 4.** *Let* $c, b \geq 1$ *and* Φ *be an implicit set system. If* Φ *is efficiently* (b, c)*-uniform then there is an algorithm that given as input* I *enumerates* $\mathcal{F}_I$ *in time* $\left(1+b-\frac{1}{c}\right)^{n+o(n)} N^{O(1)}$.

3 Random Sampling and Multivariate Subroutines

In this section, we prove Theorem 1. To do this, we first need the following lemmas.

▶ **Lemma 5.** *If* $b, c \geq 1$ *then* $b \cdot c^{\frac{1}{bc}} \leq 1 + b - \frac{1}{c}$

Proof. As both sides of the inequality are positive, it suffices to show that $\log(bc^{\frac{1}{bc}}) \leq \log(1+b-1/c)$. So we let $y = \log(1+b-1/c) - \log b - \frac{1}{bc}\log c$ and prove that $y \geq 0$ for all $b, c \geq 1$. When $c = 1$ we have that $y = 0$ for all b. We will show that for any fixed $b \geq 1$ we have that $y \geq 0$ by showing that y increases with $c \geq 1$. For fixed b, the partial derivative with respect to c is $\frac{\partial y}{\partial c} = \frac{(bc+c-1)\log c - c + 1}{bc^2(bc+c-1)}$. When $c = 1$ then for all b, $\frac{\partial y}{\partial c} = 0$. As the denominator is positive for $b, c \geq 1$ it is sufficient to show that the numerator $z = (bc + c - 1)\log c - c + 1$ is non-negative. To show that $z \geq 0$, we consider the partial derivative again with respect to c: $\frac{\partial z}{\partial c} = (b+1)\log c + b - \frac{1}{c}$ For $b, c \geq 1$, we have that $b - \frac{1}{c} \geq 0$ and $(b+1)\log(c) \geq 0$. Since $\frac{\partial z}{\partial c} \geq 0$, we conclude that z is increasing and non-negative which implies y is also increasing and non-negative, for all $b, c \geq 1$. This proves the lemma. ◀

The proof of the next lemma follows the proof of Lemma 2.2 from [5], who proved it for $b = 1$.

▶ **Lemma 6.** *Let $b, c \geq 1$, n and $k \leq n$ be non-negative integers. Then, there exists $t \geq 0$ such that*

$$\frac{\binom{n}{t}}{\binom{k}{t}} b^{n-t} c^{k-t} = \left(1 + b - \frac{1}{c}\right)^n n^{O(1)}, \quad \textit{specifically when } t = \frac{cbk - n}{cb - 1}.$$

Proof. We consider two cases. First suppose $k \leq \frac{n}{bc}$. Then for $t = 0$ the LHS (left-hand side) is at most $b^n c^k \leq b^n c^{\frac{n}{bc}} \leq (1 + b - 1/c)^n$ by Lemma 5. Now if $k > \frac{n}{bc}$ then we rewrite the LHS as

$$\frac{\binom{n}{t}}{\binom{k}{t}} b^{n-t} c^{k-t} = \frac{\binom{n}{k} b^{n-k}}{\binom{n-t}{k-t} \left(\frac{1}{bc}\right)^{k-t}}.$$

Let us lower bound the denominator. For any $x \geq 0$ and an integer $m \geq 0$,

$$\sum_{i \geq 0} \binom{m+i}{i} x^i = \sum_{i \geq 0} \binom{m+i}{m} x^i = \frac{1}{(1-x)^{m+1}}, \tag{1}$$

by a known generating function. For $m = n - k$ and $x = \frac{1}{bc}$, the summand at $i = k - t$ equals the denominator $\binom{n-t}{k-t} \left(\frac{1}{bc}\right)^{k-t}$. Since $\frac{n}{k} < bc$ we have that $\frac{m+k}{k} < \frac{1}{x}$ and the terms of this sum decay exponentially for $i > k$. Thus, the maximum term $\frac{(m+i)(m+i-1)\ldots(m+1)}{i(i-1)\ldots 1} x^i$ for this sum occurs for $i \leq k$, and its value is $\Omega\left(\left(\frac{1}{1-x}\right)^m\right)$ up to a lower order factor of $O(k)$. So by the binomial theorem the expression is at most

$$\binom{n}{k} b^{n-k} (1-x)^{n-k} n^{O(1)} = \left(1 + b - \frac{1}{c}\right)^n n^{O(1)}.$$

Specifically, the maximum term for Equation (1) occurs when $\frac{m+i}{i} = \frac{1}{x}$, that is when $\frac{n-t}{k-t} = cb$, and therefore, $t = \frac{cbk-n}{cb-1}$. ◀

▶ **Lemma 7.** *If there exist constants $b, c \geq 1$ and an algorithm for* Φ-Extension *with running time $b^{n-|X|} c^k N^{O(1)}$ then there exists a randomized algorithm for* Φ-Extension *with running time $\left(1 + b - \frac{1}{c}\right)^{n-|X|} N^{O(1)}$.*

Proof. Our proof is similar to Lemma 2.1 in [5]. Let $\mathcal{B}$ be an algorithm for Φ-Extension with running time $b^{n-|X|} c^k N^{O(1)}$. We now present a randomized algorithm $\mathcal{A}$, for the same problem for an input instance (I, X, k') with $k' \leq k$.

1. Choose an integer $t \leq k'$ depending on b, c, n, k' and $|X|$, the choice of which will be discussed later. Then select a random subset Y of $U_I \backslash X$ of size t.
2. Run Algorithm $\mathcal{B}$ on the instance $(I, X \cup Y, k' - t)$ and return the answer.

Algorithm $\mathcal{A}$ has a running time upper bounded by $b^{n-|X|-t} c^{k'-t} N^{O(1)}$. Algorithm $\mathcal{A}$ returns yes for (I, X, k') when $\mathcal{B}$ returns yes for $(I, X \cup Y, k' - t)$. In this case there exists a set $S \subseteq U_I \backslash (X \cup Y)$ of size at most $k' - t \leq k - t$ such that $S \cup X \cup Y \in \mathcal{F}_I$. This, $Y \cup S$ witnesses that (I, X, k) is indeed a yes-instance.

Next we lower bound the probability that $\mathcal{A}$ returns yes if there exists a set $S \subseteq U_I \backslash X$ of size exactly k' such that $X \cup S \in \mathcal{F}_I$. The algorithm $\mathcal{A}$ picks a set Y of size t at random from $U_I \backslash X$. There are $\binom{n-|X|}{t}$ possible choices for Y. If $\mathcal{A}$ picks one of the $\binom{k'}{t}$ subsets of S as Y then $\mathcal{A}$ returns yes. Thus, given that there exists a set $S \subseteq U_I \backslash X$ of size k' such that $X \cup S \in \mathcal{F}_I$, we have that

$$\Pr[\mathcal{A} \text{ returns yes}] \geq \Pr[Y \subseteq S] = \binom{k'}{t} / \binom{n - |X|}{t}.$$

Let $p(k') = \binom{k'}{t}/\binom{n-|X|}{t}$. For each $k' \in \{0, ..., k\}$, our main algorithm runs $\mathcal{A}$ independently $\frac{1}{p(k')}$ times with parameter k'. The algorithm returns yes if any of the runs of $\mathcal{A}$ return yes. If (I, X, k') is a yes-instance, then the main algorithm returns yes with probability at least

$$\min_{k' \leq k} \left\{ 1 - (1 - p(k'))^{\frac{1}{p(k')}} \right\} \geq 1 - \frac{1}{e} > \frac{1}{2}.$$

Next we upper bound the running time of the main algorithm, which is

$$\sum_{k' \leq k} \frac{1}{p(k')} b^{n-|X|-t} c^{k'-t} N^{O(1)} \leq \max_{k' \leq k} \frac{\binom{n-|X|}{t}}{\binom{k'}{t}} b^{n-|X|-t} c^{k'-t} N^{O(1)} \tag{2}$$

$$\leq \max_{k' \leq n-|X|} \frac{\binom{n-|X|}{t}}{\binom{k}{t}} b^{n-|X|-t} c^{k-t} N^{O(1)}. \tag{3}$$

The choice of t in algorithm $\mathcal{A}$ is chosen to minimize the value of $\frac{\binom{n-|X|}{t}}{\binom{k}{t}} b^{n-|X|-t} c^{k-t}$. For fixed n and $|X|$ the running time of the algorithm is upper bounded by

$$\max_{0 \leq k \leq n-|X|} \left\{ \min_{0 \leq t \leq k} \left\{ \frac{\binom{n-|X|}{t}}{\binom{k}{t}} b^{n-|X|-t} c^{k-t} N^{O(1)} \right\} \right\}. \tag{4}$$

By application of Lemma 6 we choose $t = \frac{cbk-(n-|X|)}{cb-1}$ to obtain the upper bound $\left(1 + b - \frac{1}{c}\right)^{n-|X|} (n - |X|)^{O(1)}$, which, combined with $n < N$, completes the proof. ◀

Running algorithm $\mathcal{A}$ with $X = \emptyset$ and for each value of $k \in \{0, ..., n\}$ results in an algorithm for Φ-Subset with running time $\left(1 + b - \frac{1}{c}\right)^n N^{O(1)}$, proving Theorem 1.

4 Derandomization

In this section we prove Theorem 2, by derandomizing the algorithm in Theorem 1.

▶ **Theorem 2.** *If there is an algorithm for* Φ-Extension *with running time* $b^{n-|X|} c^k N^{O(1)}$ *then there is an algorithm for* Φ-Subset *with running time* $(1 + b - \frac{1}{c})^{n+o(n)} N^{O(1)}$.

Given a set U and an integer $q \leq |U|$ let $\binom{U}{q}$ represent the set of sets which contain q elements of U. From [5] we define a pseudo-random object, the *set-inclusion-family*, as well as an almost optimal sub-exponential construction of these objects.

▶ **Definition 8.** Let U be a universe of size n and let $0 \leq q \leq p \leq n$. A family $\mathcal{C} \subseteq \binom{U}{q}$ is an (n, p, q)-set-inclusion family, if for every set $S \in \binom{U}{p}$, there is a set $Y \in \mathcal{C}$ such that $Y \subseteq S$.

Let $\kappa(n, p, q) = \binom{n}{q}/\binom{p}{q}$. We also make use of the following theorem.

▶ **Theorem 9** ([5])**.** *There is an algorithm that given* n, p *and* q *outputs an* (n, p, q)*-set-inclusion-family* $\mathcal{C}$ *of size at most* $\kappa(n, p, q) \cdot 2^{o(n)}$ *in time* $\kappa(n, p, q) \cdot 2^{o(n)}$.

We are now ready to prove Lemma 10, by a very similar proof to Lemma 7.

▶ **Lemma 10.** *If there exists constants* $b, c \geq 1$ *and an algorithm for* Φ-Extension *with running time* $b^{n-|X|} c^k N^{O(1)}$ *then there exists a deterministic algorithm for* Φ-Extension *with running time* $\left(1 + b - \frac{1}{c}\right)^{n-|X|} \cdot 2^{o(n)} \cdot N^{O(1)}$.

Proof. Let $\mathcal{B}$ be an algorithm for Φ-Extension with running time $b^{n-|X|}c^k N^{O(1)}$. We can then adapt Algorithm $\mathcal{A}$ from the proof of Lemma 7. Let $\mathcal{A}'$ be a new algorithm which has an input instance (I, X, k') with $k' \leq k$. Choose $t = \frac{cbk' - (n-|X|)}{cb-1}$.

1. Compute an $(n-|X|, k', t)$-set-inclusion-family $\mathcal{C}$ using the algorithm from Theorem 9 of size at most $\kappa(n-|X|, k', t) \cdot 2^{o(n)}$, in $\kappa(n-|X|, k', t) \cdot 2^{o(n)}$ time.
2. For each set Y in the set-inclusion-family $\mathcal{C}$ run algorithm $\mathcal{B}$ on the instance $(I, X \cup Y, k'-t)$ and return Yes of at least one returns Yes and No otherwise.

The running time of $\mathcal{A}'$ is upper bounded by $\kappa(n-|X|, k', t) \cdot 2^{o(n)} \cdot b^{n-|X|-t}c^{k'-t}N^{O(1)}$, a term encountered in Equation 2 with a new subexponential factor in n,

$$\max_{k' \leq k} \frac{\binom{n-|X|}{t}}{\binom{k'}{t}} \cdot b^{n-|X|-t}c^{k'-t}N^{O(1)} \cdot 2^{o(n)}.$$

From here the proof follows that of Lemma 7. ◀

The proof of Theorem 2 follows by inclusion of the factor $2^{o(n)}$.

5 Enumeration

We now proceed to prove Theorems 3 and 4 on combinatorial upper bounds and enumeration algorithms. Consider the following random process.

1. Choose an integer t based on b, c, n and k, then randomly sample a subset X of size t from U_I.
2. Uniformly at random pick a set S from $\mathcal{F}_{I,X}^{k-t}$, and output $W = X \cup S$. In the special case where $\mathcal{F}_{I,X}^{k-t}$ is empty output the empty set.

▶ **Theorem 3.** *Let $c, b \geq 1$ and Φ be an implicit set system. If Φ is (b,c)-uniform then $|\mathcal{F}_I| \leq \left(1+b-\frac{1}{c}\right)^n n^{O(1)}$ for every instance I.*

Proof. Let I be an instance, $k \leq n$. We will prove that the number of sets in $\mathcal{F}_I$ of size exactly k is upper bounded by $|\mathcal{F}_I| \leq \left(1+b-\frac{1}{c}\right)^n n^{O(1)}$, where k is chosen arbitrarily. We follow the random process described above, which picks a set W of size k from $\mathcal{F}_I$.

For each set $Z \in \mathcal{F}_I$ of size exactly k, let E_Z denote the event that the set W output in step 2 is equal to Z. We then have the following lower bound on the probability of the event E_Z:

$$\Pr[E_Z] = \Pr[X \subseteq Q \wedge S = Z \backslash X] = \Pr[X \subseteq Z] \times \Pr[S = Z \backslash Z | X \subseteq Z] = \frac{\binom{k}{t}}{\binom{n}{t}} \cdot \frac{1}{|\mathcal{F}_{I,X}^{k-t}|}$$

Since Φ is (b,c)-uniform then $\left|\mathcal{F}_{I,X}^{k-t}\right| \leq b^{n-|X|}c^k n^{O(1)}$ and X is selected such that $|X| = t$, this results in the lower bound

$$\Pr[E_Z] \geq \frac{\binom{k}{t}}{\binom{n}{t}} b^{-(n-t)}c^{-(k-t)}n^{-O(1)}.$$

A choice of t is made to minimize the lower bound, and this choice is given by Lemma 6 which states that for every $k \leq n$ there exists a $t \leq k$ such that we obtain a new lower bound

$$\Pr[E_Z] \geq \left(1+b-\frac{1}{c}\right)^{-n} \cdot n^{O(1)}$$

for every $Z \in \mathcal{F}_I$ of size k. For every individual set $Z \in \mathcal{F}_I$, the event E_Z occurs disjointly, and we have that $\sum_{Z \in \mathcal{F}_I, |Z|=k} \Pr[E_Z] \leq 1$. This fact with the lower bound of $\Pr[E_Z]$ implies an upper bound on the number of sets in $\mathcal{F}_I$ of $(1+b-\frac{1}{c})^n n^{O(1)}$, completing the proof. ◀

▶ **Theorem 4.** *Let $c, b \geq 1$ and Φ be an implicit set system. If Φ is efficiently (b, c)-uniform then there is an algorithm that given as input I enumerates $\mathcal{F}_I$ in time $\left(1 + b - \frac{1}{c}\right)^{n+o(n)} N^{O(1)}$.*

Proof. We alter the random process used to prove Theorem 3 to a deterministic one:
1. Construct a (n, k, t)-set inclusion family $\mathcal{C}$ using Theorem 6 from [5]. Loop over $X \in \mathcal{C}$.
2. For each $X \in \mathcal{C}$, loop over all sets $S \in \mathcal{F}_{I,X}^{k-t}$.

Then we output $W = X \cup S$ from these two loops. Looping over $\mathcal{C}$ instead of random sampling for X incurs a $2^{o(n)}$ overhead in the running time. As Φ is efficiently (b, c)-uniform, the inner loop requires $(1 + b - \frac{1}{c})^n N^{O(1)}$ time. In order to avoid enumerating duplicates, we save the sets which have been output in a trie and check first in linear time if a set has already been output. The product of the running times for these two nested loops results in the running time claimed by the theorem statement. ◀

6 Case Studies

This section briefly outlines case studies which used Theorem 2 and Theorem 4 in order to design faster deterministic algorithms.

6.1 Preliminaries

Let $G = (V, E)$ be a graph with a set of vertices V and a set of edges $E \subseteq \{uv : u, v \in V\}$. The *degree* $d(u)$ of a vertex u is the number of neighbors of u in G. The *degree* of a graph $\Delta(G)$ is the maximum $d(u)$ across all $u \in V$. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$ and G' is an *induced subgraph* of G if, in addition, G has no edge uv with $u, v \in V'$ but $uv \notin E'$. In this case, we also denote G' by $G[V']$. A forest is an acyclic graph. A subset $F \subseteq V$ is acyclic if $G[F]$ is a forest. An acyclic subset $F \subseteq V$ is *maximal* in G if it is not a subset of any other acyclic subset. For an acyclic subset $F \subseteq V$, we denote the set of maximal acyclic supersets of F as $\mathcal{M}_G(F)$ and the set of maximum (i.e., largest) acyclic supersets of F as $\mathcal{M}_G^*(F)$.

Let T be a subgraph of G. Define $\mathrm{Id}(T, t)$ as an operation on G which contracts all edges of T into one vertex t, removing induced loops. This may create multiedges in G. Define $\mathrm{Id}^*(T, t)$ as the operation $\mathrm{Id}(T, t)$ followed by removing all vertices connected to t by multiedges. A *non-trivial* component of a graph G is a connected component on at least two vertices. The following propositions from [6] will be useful.

▶ **Proposition 11.** *[6] Let $G = (V, E)$ be a graph, $F \subseteq V$ be an acyclic subset of vertices and T be a non-trivial component of $G[F]$. Denote by G' the graph obtained from G by the operation $Id^*(T, t)$ and let $F' = F \cup \{t\} \backslash T$. Then for $X' = X \cup \{t\} \backslash T$ where $X, X' \subseteq V$*
- *$X \in \mathcal{M}_G(F)$ if and only if $X' \in \mathcal{M}_{G'}(F')$, and*
- *$X \in \mathcal{M}_G^*(F)$ if and only if $X' \in \mathcal{M}_{G'}^*(F')$.*

Using operation Id^* on each non-trivial component of $G[F]$, results in an independent set F'.

▶ **Proposition 12.** *[6] Let $G = (V, E)$ be a graph and F be an independent set in G such that $V \backslash F = N(t)$ for some $t \in F$. Consider the graph $G' = G[N(t)]$ and for every pair of vertices $u, v \in N(t)$ having a common neighbor in $F \backslash \{t\}$ add an edge uv to G'. Denote the obtained graph by H and let $I \subseteq N(t)$. Then $F \cup I \in \mathcal{M}_G(F)$ if and only if I is a maximal independent set in H . In particular, $F \cup I \in \mathcal{M}_G^*(F)$ if and only if I is a maximum independent set in H.*

For an acyclic subset F, a so-called *active* vertex $t \in F$ and a neighbor $v \in N(t) \setminus F$, we will now define the concept of generalized neighbors of v, also known as (v)-*generalized neighbors*. Denote by K the set of vertices of F adjacent to v other than t. Let G' be the graph obtained after the operation $\mathrm{Id}(K \cup \{v\}, u)$. A vertex $w \in V(G') \setminus \{t\}$ is a (v)-generalized neighbor in G if w is a neighbor of u in G'. Denote by $\mathrm{gd}(v)$ the *generalized degree* of v which is the number of (v)-generalized neighbors for a given v.

6.2 Feedback Vertex Set

First we describe the extension variant of Feedback Vertex Set

Feedback Vertex Set Extension	
Input:	A graph $G = (V, E)$, vertex subset $X \subseteq V$ and an integer k
Question:	Does there exist subset $S \subseteq V \setminus X$ such that $S \cup X$ is a FVS and $\lvert S \rvert \leq k$?

Instead of directly finding the feedback vertex set in a graph, we present algorithm $\texttt{mif}(G, F, k)$ [6] which computes for a given graph G and an acylic set F the maximum size of an induced forest F' containing F with $|F'| \geq n - k$. This means that $G - F$ is a minimal feedback vertex set of size at most k. This algorithm can easily be turned into an algorithm computing at least one such set.

During the execution of `mif` one vertex $t \in F$ is called an *active vertex*. Algorithm `mif` then branches on a chosen neighbor of t. Let $v \in N(t)$. Let k be the set of all vertices of $F \setminus \{t\}$ that are adjacent to v and parameter k which represents a bound on the size of the feedback vertex set.

As well as describing the algorithm we simultaneously perform the running time analysis which uses the Measure and Conquer framework and Lemma 13 at its core.

▶ **Lemma 13.** *[10] Let A be an algorithm for a problem P, B be an algorithm for a class C of instances of P, $c \geq 0$ and $r > 1$ be constants, and $\mu(\cdot), \mu_B(\cdot), \eta(\cdot)$ be measures for P, such that for any input instance I from C, $\mu_B(\cdot) \leq \mu(I)$, and for any input instance I, A either solves P on $I \in C$ by invoking B with running time $O(\eta(I)^{c+1} r^{\mu B(I)})$, or reduces I to k instances $I_1, ..., I_k$, solves these recursively, and combines their solutions to solve I, using time $O(\eta(I)^c)$ for the reduction and combination steps (but not the recursive solves),*

$$(\forall i) \quad \eta(I_i) \leq \eta(I) - 1, \qquad \text{and} \qquad \sum_{i=1}^{k} r^{\mu(I_i)} \leq r^{\mu(I)}. \tag{5}$$

Then A solves any instance I in time $O(\eta(I)^{c+1} r^{\mu(I)})$.

Branching constraints of the form $\sum_{i=1}^{j} 2^{-\delta_i} \leq 1$ are given as branching vectors $(\delta_1, ..., \delta_j)$.

6.2.1 Measure

To upper bound the exponential time complexity of the algorithm `mif` we use the measure

$$\mu = |N(t) \setminus F| w_1 + |V \setminus (F \cup N(t))| w_2 + k \cdot w_k.$$

In other words, each vertex in F has weight 0, each vertex in $N(t)$ has weight w_1, each other vertex has weight w_2 and each unit of budget for the feedback vertex set has weight w_k, in the measure with an active vertex t.

6.2.2 Algorithm

The description of `mif` consists of a sequence of cases and subcases. The first case which applies is used, and inside a given case the hypotheses of all previous cases are assumed to be false. Preprocessing procedures come before main procedures.

Preprocessing

1. If G consists of $j \geq 2$ connected components $G_1, G_2, ..., G_j$, then the algorithm is called on each component. For $F_i = G_i \cap F$ for all $i \in \{1, 2, ..., j\}$ and $\sum_{i=1}^{j} k_i \leq k$ then

$$\texttt{mif}(G, F, k) = \sum_{i=1}^{j} \texttt{mif}(G_i, F_i, k_i)\,.$$

2. If F is not independent, then apply operation $\mathrm{Id}^*(T, v_T)$ on an arbitrary non-trivial component T of F. If T contains the active vertex then v_T becomes active. Let G' be the resulting graph and let F' be the set of vertices of G' obtained from F. Then

$$\texttt{mif}(G, F, k) = \texttt{mif}(G', F', k) + |T| - 1\,.$$

Main Procedures

1. If $k < 0$ then $\texttt{mif}(G, F, k) = 0$.
2. If $F = \emptyset$ and $\Delta(G) \leq 1$ then $\mathcal{M}_G(F) = \{V\}$ and $\texttt{mif}(G, F, k) = |V|$.
3. If $F = \emptyset$ and $\Delta(G) \geq 2$ then the algorithm chooses a vertex $t \in G$ of degree at least 2. Then t is either contained in a maximum induced forest or not. The algorithm branches on two subproblems and returns the maximum:

$$\texttt{mif}(G, F, k) = \max\{\texttt{mif}(G, F \cup \{t\}, k), \texttt{mif}(G \backslash \{t\}, F, k-1)\}.$$

   The first branch reduces the weight of t to zero, as it is in F, and at least 2 neighbors have a reduced degree from w_2 to w_1. In the second branch we remove t from the graph, meaning it will be in the feedback vertex set. We thus also gain a reduction of w_k in the measure. Hence this rule induces the branching constraint

$$(w_2 + 2(w_2 - w_1), w_2 + w_k).$$

4. If F contains no active vertex then choose an arbitrary vertex $t \in F$ as an active vertex. Denote the active vertex by t from now on.
5. If $V \backslash F = N(t)$ then the algorithm constructs the graph H from Proposition 12 and computes a maximum independent set I in G of maximum size $n - k$. Then

$$\texttt{mif}(G, F, k) = |F| + |I|.$$

6. If there is $v \in N(t)$ with $\mathrm{gd}(v) \leq 1$ then add v to F.

$$\texttt{mif}(G, F, k) = \texttt{mif}(G, F \cup \{v\}, k).$$

7. If there is $v \in N(t)$ with $\mathrm{gd}(v) \geq 4$ then either add v to F or remove v from G.

$$\texttt{mif}(G, F, k) = \max\{\texttt{mif}(G, F \cup \{v\}, k), \texttt{mif}(G \backslash \{v\}, F, k-1)\}.$$

   The first case adds v to F reducing the measure by w_1, and a minimum of $4(w_2 - w_1)$ for all the (v)-generalized neighbors. The other case removes v this decreasing the measure by w_k. Hence this rule induces the branching constraint

$$(w_1 + 4(w_2 - w_1), w_1 + w_k).$$

8. If there is $v \in N(t)$ with $\text{gd}(v) = 2$ then denote the (v)-generalized neighbors by u_1 and u_2. Either add v to F or remove v from G but add u_1 and u_2 to F. If adding u_1 and u_2 to F induces a cycle, we just ignore the last branch.

$$\texttt{mif}(G, F, k) = \max\{\texttt{mif}(G, F \cup \{v\}, k), \texttt{mif}(G \backslash \{v\}, F \cup \{u_1, u_2\}, k-1)\}.$$

Let $i \in \{0, 1, 2\}$ be the number of vertices adjacent to v with weight w_2. The first case adds v to F, and hence all i w_2-weight neighbors of v reduce to w_1, and the other $2-i$ vertices of weight w_1 induce a cycle, hence we remove them from G and reduce the measure by $(2-i)w_k$. The second case removes v and adds both u_1 and u_2 to F. This causes a reduction of iw_2 for the relevant vertices and $(2-i)w_1$ for the other vertices, and a single w_k reduction due to the removal of v. This rule induces the branching constraint

$$(w_1 + i(w_2 - w_1) + (2-i)w_1 + (2-i)w_k, w_1 + iw_2 + (2-i)w_1 + w_k).$$

9. If all vertices in $N(t)$ have exactly three generalized neighbors then at least one of these vertices must have a generalized neighbor outside $N(t)$, since the graph is connected and the condition of the case Main 6 does not hold. Denote such a vertex by v and its generalized neighbors by u_1, u_2 and u_3 in such a way that $u_1 \notin N(t)$. Then we either add v to F; or remove v from G but add u_1 to F; or remove v and u_1 from G and add u_2 and u_3 to F. Similar to the previous case, if adding u_2 and u_3 to F induces a cycle, we just ignore the last branch.

$$\begin{aligned}\texttt{mif}(G, F) = \max\{&\texttt{mif}(G, F \cup \{v\}, k), \texttt{mif}(G \backslash \{v\}, F \cup \{u_1\}, k-1),\\ &\texttt{mif}(G \backslash \{v, u_1\}, F \cup \{u_2, u_3\}, k-2)\}.\end{aligned}$$

Let $i \in \{1, 2, 3\}$ be the number of vertices adjacent to v with weight w_2. The first and last cases are analogous to the analysis done in Main 8. The second case removes v from the forest hence adding it to the minimum feedback vertex set and reducing the measure by $w_1 + w_k$. A reduction of w_2 is gained by adding u_1 to F. Then this rule induces the branching constraint

$$(w_1 + i(w_2 - w_1) + (3-i)w_1 + (3-i)w_k, w_1 + w_2 + w_k, w_1 + iw_2 + (3-i)w_1 + 2w_k).$$

6.2.3 Results

▶ **Theorem 14.** *Let G be a graph on n vertices. Then a minimal feedback vertex set in G can be found in time $O(1.7117^n)$.*

Proof. Using the algorithm above along with the measure μ, the following values of weights can be shown to satisfy all the branching vector constraints generated above.

$$w_1 = 0.2775 \qquad w_2 = 0.6250 \qquad w_k = 0.2680$$

These weights result in an upper bound for the running time of `mif` as $O(1.5422^n \cdot 1.2041^k)$ for computing a maximally induced forest of size a least $n-k$, and hence we have the running time for Feedback Vertex Set Extension of $O(1.5422^{n-|X|} \cdot 1.2041^k)$. By Theorem 2 this results in a $O(1.7117^n)$ algorithm for computing a minimal feedback vertex set. ◀

6.3 Minimal Vertex Covers

The following result on minimal vertex covers of size at most k is needed in the next section.

▶ **Theorem 15.** *Let γ be a constant with $0.169925 \approx 2\log_2 3 - 3 \leq \gamma \leq 1$. For every $n \geq k \geq 0$, and every graph G on n vertices, the number of minimal vertex covers of size at most k of G is at most $2^{\beta n+\gamma k}$, where $\beta = (1-\gamma)/2$.*

For $\gamma = \frac{1}{3}$, this implies that G has at most $2^{(n+k)/3}$ minimal vertex covers of size at most k.

6.4 Minimal Feedback Vertex Sets

We apply a similar methodology to enumerating minimal feedback vertex sets on an undirected graph with n vertices. The algorithm is similar in construction to one used in [6] yet a large amount of case analysis was added, along with potential functions in combination with the Measure and Conquer framework.

▶ **Theorem 16.** *For a graph G with n vertices, all minimal feedback vertex sets can be enumerated in time $O(1.8527^n)$.*

6.5 Minimal Hitting Sets

Based on [2] we once again apply a multivariate analysis to enumerating all minimal hitting sets on a hypergraph of rank 3.

▶ **Theorem 17.** *For a hypergraph H with n vertices and rank 3, all minimal hitting sets can be enumerated in time $O(1.6627^n)$.*

7 Conclusion

The main contribution of this paper is a framework allowing us to turn many $b^n c^k N^{O(1)}$ time algorithms for subset and subset enumeration problems into $(1 + b - \frac{1}{c})^n N^{O(1)}$ time algorithms, generalizing a recent framework of Fomin et al. [5].

The main complications in using the framework are, firstly, that new algorithms or running-time analyses are often needed, and, secondly, that such analyses need solutions to non-convex programs in the Measure and Conquer framework. In the usual Measure and Conquer analyses [7],the objective is to upper bound a single variable (α) which upper bounds the exponential part of the running time ($2^{\alpha n}$) subject to convex constraints. Thus, it is sufficient to solve a convex optimization problem to minimize the running time [10, 12] resulting from the constraints derived from the analysis. Here, the objective function ($2^{\alpha} - 2^{-w_k}$) is non-convex. While experimenting with a range of solvers (Couenne, IPOPT, MINOS, SNOPT), either guaranteeing to find a global optimum (slow and used for optimality checks) or only a local optimum (faster and used mainly in the course of the algorithm design), we experienced on one hand that the local optima found by solvers are often the global optimum, but on the other hand that weakening non-tight constraints can sometimes lead to a better globally optimum solution.

Acknowledgements. We thank Daniel Lokshtanov, Fedor V. Fomin, and Saket Saurabh for discussions inspiring some of this work.

References

1 Jesper Makholm Byskov. Enumerating maximal independent sets with applications to graph colouring. *Oper. Res. Lett.*, 32(6):547–556, 2004. doi:10.1016/j.orl.2004.03.002.

2 Manfred Cochefert, Jean-François Couturier, Serge Gaspers, and Dieter Kratsch. Faster algorithms to enumerate hypergraph transversals. In *Latin American Symposium on Theoretical Informatics*, pages 306–318. Springer, 2016.

3 Rodney G. Downey and Michael R. Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

4 David Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms Appl.*, 7(2):131–140, 2003.

5 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2016)*, pages 764–775. ACM, 2016. doi:10.1145/2897518.2897551.

6 Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the minimum feedback vertex set problem: Exact and enumeration algorithms. *Algorithmica*, 52(2):293–307, 2008.

7 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56(5), 2009. doi:10.1145/1552285.1552286.

8 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010. An EATCS Series: Texts in Theoretical Computer Science.

9 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and cmso. *SIAM Journal on Computing*, 44(1):54–87, 2015.

10 Serge Gaspers. *Exponential Time Algorithms – Structures, Measures, and Bounds*. VDM, 2010. URL: http://amzn.com/3639218256.

11 Serge Gaspers and Edward Lee. Exact algorithms via multivariate subroutines, April 2017. arXiv e-prints. URL: https://arxiv.org/abs/1704.07982, arXiv:1704.07982.

12 Serge Gaspers and Gregory B. Sorkin. A universally fastest algorithm for Max 2-Sat, Max 2-CSP, and everything in between. *Journal of Computer and System Sciences*, 78(1):305–335, 2012. doi:10.1016/j.jcss.2011.05.010.

13 Igor Razgon. Exact computation of maximum induced forest. In *Scandinavian Workshop on Algorithm Theory*, pages 160–171. Springer, 2006.

14 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. In *International Symposium on Algorithms and Computation*, pages 328–338. Springer, 2013.

Exploring the Complexity of Layout Parameters in Tournaments and Semi-Complete Digraphs*†

Florian Barbero[1], Christophe Paul[2], and Michał Pilipczuk[3]

1 **LIRMM, Université de Montpellier, Montpellier, France**
florian.barbero@lirmm.fr
2 **LIRMM, CNRS, Université de Montpellier, Montpellier, France**
christophe.paul@lirmm.fr
3 **University of Warsaw, Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl

Abstract

A simple digraph is *semi-complete* if for any two of its vertices u and v, at least one of the arcs (u, v) and (v, u) is present. We study the complexity of computing two layout parameters of semi-complete digraphs: cutwidth and optimal linear arrangement (Ola). We prove that:

- Both parameters are NP-hard to compute and the known exact and parameterized algorithms for them have essentially optimal running times, assuming the Exponential Time Hypothesis.
- The cutwidth parameter admits a quadratic Turing kernel, whereas it does not admit any polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. By contrast, Ola admits a linear kernel.

These results essentially complete the complexity analysis of computing cutwidth and Ola on semi-complete digraphs. Our techniques can be also used to analyze the sizes of minimal obstructions for having small cutwidth under the induced subdigraph relation.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases cutwidth, OLA, tournament, semi-complete digraph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.70

1 Introduction

A directed graph (digraph) is *simple* if it does not contain a self-loop or multiple arcs with the same head and tail. A simple digraph is *semi-complete* if for any pair of its vertices u and v, at least one of the arcs (u, v) or (v, u) is present. If moreover exactly one of them is present for each pair u, v, then a semi-complete digraph is called a *tournament*. Tournaments and semi-complete digraphs form a rich and interesting subclass of directed graphs; we refer to the book of Bang-Jensen and Gutin [1] for an overview.

We study two layout parameters for tournaments and semi-complete digraphs: cutwidth and optimal linear arrangement (Ola). Suppose π is an ordering of the vertices of a digraph D. With each prefix of π we associate a *cut* defined as the set of arcs with head in the prefix and tail outside of it. The *width* of π is defined as the maximum size among the cuts associated with the prefixes of π. The *cutwidth* of D, denoted $\mathsf{ctw}(D)$, is the minimum width among orderings of the vertex set of D. Optimal linear arrangement (Ola) is defined

* A full version of the paper is available at https://arxiv.org/abs/1706.00617.

† The research of F. Barbero and C. Paul is supported by the project DE-MO-GRAPH ANR-16-CE40-0028, and the research of Mi. Pilipczuk is supported by Polish National Science Centre grant UMO-2013/11/D/ST6/03073. Mi. Pilipczuk is also supported by the Foundation for Polish Science via the START stipend programme.

similarly, but when defining the width of π, called in this context the *cost* of π, we take the sum of the cutsizes associated with prefixes, instead of the maximum. Then the Ola-cost of a digraph D, denoted $\textsc{Ola}(D)$, is the minimum cost among vertex orderings of D.

Known results. The study of cutwidth in the context of tournaments and semi-complete digraphs started with the work of Chudnovsky, Fradkin, and Seymour [3, 4, 10], who identified this layout parameter as the right dual notion to immersions in semi-complete digraphs. In particular, it is known that excluding a fixed digraph as an immersion yields a constant upper bound on the cutwidth of a semi-complete digraph [3, 16]. Due to this connection, cutwidth played a pivotal role in the proof of Chudnovsky and Seymour that the immersion order is a well quasi-order on tournaments [4].

The algorithmic properties of cutwidth were preliminarily investigated by Chudnovsky, Fradkin, and Seymour [3, 10, 9]. In Fradkin's PhD thesis [9], several results on the tractability of computing the cutwidth are presented. In particular, it is shown that the cutwidth of a tournament can be computed optimally by just sorting vertices according to their outdegrees, whereas in semi-complete digraphs a similar approach yields a polynomial-time 2-approximation algorithm. The problem becomes NP-hard on super-tournaments, that is, when multiple parallel arcs are allowed. Later, the third author together with Fomin proposed a parameterized algorithm for computing the cutwidth of a semi-complete digraph with running time $2^{\mathcal{O}(\sqrt{k \log k})} \cdot n^2$ [8, 17], where n is the number of vertices and k is the target width. Using the same techniques, Ola in semi-complete digraphs can be solved in time $2^{\mathcal{O}(k^{1/3}\sqrt{\log k})} \cdot n^2$ [8, 17], where k is the target cost. It was left open whether the running times of these parameterized algorithms are optimal [17]. In fact, even settling the NP-hardness of computing cutwidth and Ola in semi-complete digraphs was open [9, 17].

Our contribution. We study two aspects of the computational complexity of computing cutwidth and Ola of semi-complete digraphs: optimality of parameterized algorithms and kernelization. First, we prove that these problems are NP-hard and we provide almost tight lower bounds for the running times of algorithms solving them, based on the Exponential Time Hypothesis (ETH). Second, we describe the kernelization complexity of the two parameters in semi-complete digraphs. In particular, we show, somewhat surprisingly, that the problem of computing the cutwidth admits a quadratic Turing kernel, while the existence of a classic polynomial kernel would imply that $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. The proofs of these main results yield complementary algorithmic and structural results that we discuss later.

Our algorithmic lower bounds are encapsulated in the following theorem.

▶ **Theorem 1.** *For semi-complete digraphs, both computing the cutwidth and computing the* Ola*-cost are* NP*-hard problems. Moreover, unless the Exponential Time Hypothesis fails:*

- *the cutwidth cannot be computed in time $2^{o(n)}$ nor in time $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$; and*
- *the* Ola*-cost cannot be computed in time $2^{o(n)}$ nor in time $2^{o(k^{1/3})} \cdot n^{\mathcal{O}(1)}$.*

Here, n is the vertex count of the input semi-complete digraph, and k is the target width/cost.

Thus, Theorem 1 shows that the known parameterized algorithms of Fomin and Pilipczuk [8] are optimal under ETH, up to $\sqrt{\log k}$ factor in the exponent. Note that both cutwidth and Ola can be computed in time $2^n \cdot n^{\mathcal{O}(1)}$ using standard dynamic programming on subsets, so we obtain tight lower bounds also for exact exponential-time algorithms.

Next, we turn our attention to kernelization. Recall that a *kernelization algorithm* (or *kernel*, for short) is a polynomial-time algorithm that given some instance of a parameterized problem, returns an equivalent instance whose size is bounded by a computable function of

the input parameter; this function is called the *size* of the kernel. We are mostly interested in finding polynomial kernels, as admitting a kernel of any computable size is equivalent to fixed-parameter tractability of the problem [6, 5]. Consider the parameterized problems of deciding whether a given semi-complete digraph has cutwidth, respectively Ola-cost, bounded by a given integer c, which is considered to be the parameter. As shown by the next two theorems, the kernelization complexity of these two problems is quite different.

▶ **Theorem 2.** *There exists a polynomial-time algorithm that given an arbitrary digraph D and an integer c, either correctly concludes that $\textsc{Ola}(D) > c$, or finds a digraph D' on at most $2c$ vertices such that $\textsc{Ola}(D') = \textsc{Ola}(D)$.*

▶ **Theorem 3.** *Unless $\mathsf{NP} \subseteq \mathsf{coNP}/poly$, there exists no polynomial-size kernelization algorithm for the problem of computing the cutwidth of a semi-complete digraph.*

The proofs of these two theorems directly follow from the understanding of the contribution of strongly connected components in optimal orderings. On one side, the contribution to the Ola-cost of each strongly connected component is at least linear in its size, implying Theorem 2. On the other side, we can observe that the cutwidth of a digraph is the maximum over the cutwidth of its strongly connected components, which implies that, like many other width parameters, cutwidth is an AND-*composable* parameter [7, 5].

However, an alternative notion of kernelization, called *Turing kernelization*, has been also studied intensively in the literature; cf. the discussion in [5]. In this framework, it is not required that the instance at hand is reduced to one equivalent small instance, but rather that the whole problem can be solved in polynomial time assuming oracle access to an algorithm solving instances of size bounded by a function of the parameter. Somewhat surprisingly, we prove that the problem of computing the cutwidth of a semi-complete digraph admits a quadratic Turing kernel, which is encapsulated in the following theorem.

▶ **Theorem 4.** *There exists a polynomial-time algorithm that given a semi-complete digraph D and integer c, either correctly concludes that $\mathsf{ctw}(D) > c$ or outputs a list of at most n induced subdigraphs $D_1, \ldots, D_\ell$ of D, each with at most $\mathcal{O}(c^2)$ vertices, such that $\mathsf{ctw}(D) \leq c$ if and only if $\mathsf{ctw}(D_i) \leq c$ for each $i \in \{1, 2, \ldots, \ell\}$.*

Theorem 4 gives a so-called AND-*Turing kernel*, meaning that the algorithm just computes the output list without any oracle calls, and the answer to the input instance is the conjunction of the answers to the output small instances. This places the problem of computing the cutwidth of a semi-complete digraph among very few known examples of natural problems where classic and Turing kernelization have different computational power [2, 15, 11, 18, 21]. Moreover, this is the first known to us polynomial AND-Turing kernel for a natural problem: examples of Turing kernelization known in the literature are either OR-Turing kernels [2, 11, 18], or adaptative kernels that fully exploit the oracle model [15, 21]. As separating classic and Turing kernelization is arguably one of the most important complexity-theoretical open problems within parameterized complexity [6, 13, 5], we find this new example intriguing.

As a byproduct of our approach to proving Theorem 4, we obtain also polynomial upper bounds on the sizes of minimal obstructions to having small cutwidth. More precisely, for a positive integer c, a digraph D is called *c-cutwidth-minimal* if the cutwidth of D is at least c, but the cutwidth of every proper induced subdigraph of D is smaller than c.

▶ **Theorem 5.** *For every positive integer c, every c-cutwidth-minimal semi-complete digraph has at most $\mathcal{O}(c^2)$ vertices.*

▶ **Theorem 6.** *For every positive integer c, every c-cutwidth-minimal tournament has at most $2c + 2\lceil\sqrt{2c}\rceil + 1$ vertices.*

The bound of Theorem 6 is almost tight, as there exist c-cutwidth-minimal tournaments with $2c+1$ vertices. Theorems 5 and 6 have direct algorithmic applications for parameterized graph modification problems related to cutwidth, e.g., c-Cutwidth Vertex Deletion: remove at most k vertices from a given digraph to obtain a digraph of cutwidth at most c.

Approach. The starting point of our study is the approach used in the earlier works by Fradkin [9] and by the third author [17, 16], namely to sort the vertices of the given semi-complete digraph according to non-decreasing indegrees, and argue that this ordering has to resemble an optimum one. As shown by Fradkin [9], this statement may be made precise for tournaments: any indegree ordering has optimum cutwidth.

We present a somewhat finer study of this argument using the notion of a *minimum ordering*. Namely, a vertex ordering π of a digraph D is *minimum* if for any other vertex ordering π' of D and any $i \in \{1, 2, \ldots, n-1\}$, the cutsize in π between the prefix of length i and the complementary suffix is smaller or equal than the cutsize defined in the same manner in π'. The sorting argument of Fradkin [9] in fact yields the following: a vertex ordering of a tournament is minimum if and only if it is sorted according to indegrees. In particular, every tournament admits a minimum ordering, computable in polynomial time. Since every minimum ordering optimizes both the cutwidth and the Ola-cost, we obtain the following.

▶ **Theorem 7.** *The cutwidth and* Ola *of a tournament can be computed in polynomial time.*

Unfortunately, general semi-complete digraphs may not admit minimum orderings. However, a semi-complete digraph can be relaxed to a *fractional tournament* with a loss of factor 2 on the cutwidth and the Ola-cost. The ordering argument for tournaments may be applied to fractional tournaments as well, and thus we obtain a polynomial-time 2-approximation.

▶ **Theorem 8.** *There exists a polynomial-time algorithm that given a semi-complete digraph D, outputs an ordering of its vertices of width, and respectively cost, upper bounded by twice the cutwidth, respectively* Ola*-cost, of D.*

While Theorems 7 and 8 for cutwidth were already proved by Fradkin [9], the applicability of the approach to Ola is a new contribution of this work. We choose to include the proofs of Theorems 7 and 8 in this work for two reasons. First, the fine understanding of minimum orderings is a basic tool needed in the proofs of our main results. Second, the abovementioned results of Fradkin [9] were communicated only in her PhD thesis and, to the best of our knowledge, were neither included in any published work, nor we have found any reference to them. We believe that these fundamental observations deserve a better publicity.

For the proof of Theorem 1, we construct a reduction from NAE-3SAT, a variant of 3SAT. In case the input formula is satisfiable, the output semi-complete digraph admits a minimum vertex ordering with a precisely specified vector of cutsizes. On the other hand, admitting any ordering of width bounded by the maximum of these cutsizes implies satisfiability of the input formula. Thus, the same reduction may serve to certify the hardness of both computing the cutwidth and computing the Ola-cost of a semi-complete digraph.

For the proof of Theorem 4, we use the notion of a *lean ordering*; see e.g. [12, 20]. Intuitively, a vertex ordering is lean if it is tight with respect to cut-flow duality: there are systems of arc-disjoint paths which certify that cutsizes along the ordering cannot be improved. Lean orderings and decompositions are commonly used in the analysis of obstructions for various width notions, as well as for proving well quasi-order results. In particular, the concept of a lean ordering for cutwidth of digraphs was used by Chudnovsky and Seymour in their proof that the immersion order is a well quasi-order on tournaments [4].

Lean orderings are used in the proof of Theorem 4 as follows. We first compute a 2-approximate ordering using Theorem 8, and then we exhaustively improve it until it becomes lean. Let σ be the obtained ordering, and consider the sequence of cutsizes along σ. The next observation is crucial. Due to leanness, if some cutsize in this sequence is smaller or equal than $\Omega(c)$ cutsizes to the left and to the right, then there is some optimum-width ordering that uses the corresponding cut; that is, the prefix of σ up to this cut is also a prefix of some optimum-width ordering. We call such cuts *milestones*. It is not hard to prove that a milestone can be found every $\mathcal{O}(c^2)$ vertices in the ordering σ. Thus we are able to partition the digraph into pieces of size $\mathcal{O}(c^2)$ that may be treated independently. Each of these pieces gives rise to one digraph D_i in the output of the kernelization algorithm.

Theorem 5 follows easily from Theorem 4; basically, the algorithm applied to a c-cutwidth minimal semi-complete digraph cannot output only smaller digraphs. For Theorem 6 we use our finer understanding of minimum orderings in tournaments. We remark that from the well-quasi order result of Chudnovsky and Seymour [4], it follows that the number of minimal *immersion* obstructions for tournaments of cutwidth at most c is finite. However, this holds only for tournaments, yields a non-explicit upper bound on obstruction sizes, and applies to immersion and not induced subdigraph obstructions.

Organization. In Section 2 we introduce notation, recall basic definitions, and prove Theorems 7 and 8. In Sections 3 and 4 we prove Theorems 4 and 1, respectively. The proofs of statements marked with ♠ will appear in the full version of the paper.

2 Preliminaries and basic results

Notation. We use standard graph notation for digraphs. All digraphs considered in this paper are simple, i.e., they do not contain a self-loop or multiple arcs with the same head and tail. For definitions of tournaments and semi-complete digraphs, see the first paragraph of Section 1. If present in a digraph, the arcs (u, v) and (v, u) are called *symmetric arcs*.

For two integers $p \leq p'$, let $[p, p'] \subseteq \mathbb{Z}$ be the set of integers between p and p'. If $p < p'$, we set $[p', p] = \emptyset$ by convention. A *vertex ordering* of a digraph D is a bijective mapping $\pi \colon V(D) \to [1, n]$, where $n = |V(D)|$. A vertex $u \in V(D)$ is *at position* i *in* π if $\pi(u) = i$. We denote this unique vertex by π_i. The *prefix* of length i of π is $\pi_{\leq i} = \{\pi_j \colon j \in [1, i]\}$; we set $\pi_{\leq i} = \emptyset$ when $i \leq 0$, and $\pi_{\leq i} = V(D)$ when $n \leq i$. We extend this notation to prefixes and suffixes of orderings naturally, e.g., $\pi_{>i} = V(D) \setminus \pi_{\leq i}$ is the set of the last $n - i$ vertices in π. The notions of restriction and concatenation of ordering(s) are defined naturally.

An arc $(\pi_i, \pi_j) \in E(D)$ is a *feedback arc for* π if $i > j$, that is, if π_i is after π_j in π. Given a digraph $D = (V, E)$, an ordering π of V and an integer i, we define the *cut* E^i_π as the set of feedback arcs $E(\pi_{>i}, \pi_{\leq i})$. The tuple $\mathsf{cuts}\langle D, \pi\rangle = (|E^0_\pi|, |E^1_\pi|, \ldots, |E^n_\pi|)$ is called the *cut vector* of π, and we denote $\mathsf{cuts}\langle D, \pi\rangle(i) = |E^i_\pi|$. Let $\preceq$ be the product order on tuples: for n-tuples A, B, we have $A \preceq B$ iff $A(i) \leq B(i)$ for all $i \in [0, n]$. We define $A \prec B$ as $A \preceq B$ and $A \neq B$. We say that a vertex ordering π is *minimum for* D if for all vertex orderings π' of D we have $\mathsf{cuts}\langle D, \pi\rangle \preceq \mathsf{cuts}\langle D, \pi'\rangle$. Note that a minimum vertex ordering may not exist.

The *width* of a vertex ordering π of a digraph D, denoted $\mathsf{ctw}(D, \pi)$, is equal to $\max\{\mathsf{cuts}\langle D, \pi\rangle\}$, where max on a tuple yields the largest coordinate. The *cutwidth* of D, denoted $\mathsf{ctw}(D)$, is the minimum width among vertex orderings of D. Similarly, the *cost* of π, denoted $\textsc{Ola}(D, \pi)$, is equal to $\sum\{\mathsf{cuts}\langle D, \pi\rangle\}$, where $\sum$ on a tuple yields the sum of coordinates. This is equivalent to summing $j - i$ for all feedback arcs (π_j, π_i) in π. The

OLA-cost of D, denoted $\textsc{Ola}(D)$, is the minimum cost among vertex orderings of D. A vertex ordering π of D satisfying $\mathsf{ctw}(D) = \mathsf{ctw}(D, \pi)$, or $\textsc{Ola}(D) = \textsc{Ola}(D, \pi)$, is respectively called ctw-*optimal* or $\textsc{Ola}$-*optimal* for D. Note that a minimum ordering for D, if existent, is always ctw-*optimal* and $\textsc{Ola}$-*optimal* for D.

Exponential-Time Hypothesis. The *Exponential Time Hypothesis* (ETH) of Impagliazzo et al. [14] states that for some constant $c > 0$, there is no algorithm for 3SAT that would run in time $2^{cn} \cdot (n+m)^{\mathcal{O}(1)}$, where n and m are the numbers of variables and clauses of the input formula, respectively. Using the *Sparsification Lemma* [14] one can show that under ETH, there is a constant $c > 0$ such that 3SAT cannot be solved in time $2^{cm} \cdot (n+m)^{\mathcal{O}(1)}$. In this work we use the NAE-3SAT problem (for Not-All-Equal), which is a variant of 3SAT where a clause is considered satisfied only when at least one, but not all of its literals are satisfied. Schaefer [19] gave a linear reduction from 3SAT to NAE-3SAT, which immediately yields:

▶ **Corollary 9.** *Unless ETH fails,* NAE-3SAT *cannot be solved in time* $2^{o(m)} \cdot (n+m)^{\mathcal{O}(1)}$, *where* n *and* m *are the numbers of variables and clauses of the input formula, respectively.*

Theorems 7 and 8. We now proceed to proving Theorems 7 and 8; recall that for cutwidth, these results have been already established by Fradkin [9]. However, we use this opportunity to present the reasoning in a more insightful manner and more general context, which also yields a better combinatorial understanding that will be helpful later. The core idea is to work in a more general setting of linear relaxations of tournaments, as defined next.

A *fractional tournament* is a pair $T = (V, \omega)$, where V is a finite vertex set and $\omega \colon V^2 \to \mathbb{R}_{\geq 0}$ is a weight function that satisfies the following properties: $\omega(u,u) = 0$ for all $u \in V$, and $\omega(u,v) + \omega(v,u) = 1$ for all pairs of different vertices u, v. Thus, by requiring the weights to be integral we recover the original definition of a tournament. We extend the notation for digraphs to fractional tournaments as follows. For $X, Y \subseteq V$ we define $\omega(X,Y) = \sum_{x \in X,\, y \in Y} \omega(x,y)$, and for $u \in V$ we define $\omega^-(u) = \omega(V, \{u\})$ and $\omega^+(u) = \omega(\{u\}, V)$. The notions of (minimum) vertex orderings, cut vectors, cutwidth, and OLA-cost are extended naturally: the cardinality of any cut $E(X,Y)$ is replaced by the sum of weights $\omega(X,Y)$.

Suppose $T = (V, \omega)$ is a fractional tournament. We say that a vertex ordering π of T is *sorted* if for any pair of different vertices u and v, if $\omega^-(u) < \omega^-(v)$, then $\pi(u) < \pi(v)$; in other words, the vertices are sorted according to their indegrees. The following lemma encapsulates the essence of our approach.

▶ **Lemma 10 (♠).** *A vertex ordering of a fractional tournament is minimum iff it is sorted.*

The proof of Theorem 7, even in the more general setting of fractional tournaments, is now immediate. We just sort the vertices according to their indegrees ω^-. By Lemma 10, the obtained ordering is minimum, hence it is both ctw-optimal and $\textsc{Ola}$-optimal.

Lemma 10 cannot be generalized to the semi-complete setting, as there are semi-complete digraphs that do not admit any minimum ordering.

We now give a 2-approximation algorithm for general semi-complete digraphs. The main idea is to relax a given semi-complete digraph to a fractional tournament. Precisely, for a semi-complete digraph D, consider its *relaxation* T_D which is a fractional tournament on the vertex set $V(D)$, where for every pair of different vertices u and v, we put

- $\omega(u,v) = 1$ and $\omega(v,u) = 0$, when (u,v) is present in D but (v,u) is not present; and
- $\omega(u,v) = \omega(v,u) = 1/2$, when (u,v) and (v,u) is a pair of symmetric arcs in D.

We put $\omega(u,u) = 0$ for every vertex u, thus T_D is indeed a fractional tournament. Observe that for any pair of vertices u, v, we have $|E(\{u\},\{v\})|/2 \leq \omega_{T_D}(\{u\},\{v\}) \leq |E(\{u\},\{v\})|$.

Therefore, for every vertex ordering π of D and every index $i \in [0, n]$, it holds that

$$\mathsf{cuts}\langle D, \pi\rangle(i)/2 \leq \mathsf{cuts}\langle T_D, \pi\rangle(i) \leq \mathsf{cuts}\langle D, \pi\rangle(i).$$

In particular we have $\mathsf{ctw}(D)/2 \leq \mathsf{ctw}(T_D) \leq \mathsf{ctw}(D)$ and $\textsc{Ola}(D)/2 \leq \textsc{Ola}(T_D) \leq \textsc{Ola}(D)$. The proof of Theorem 8 is now immediate: just output any sorted ordering of T_D.

3 Turing kernel

In this section we prove Theorem 4, that is, we give a quadratic Turing kernel for the problem of computing the cutwidth of a semi-complete digraph. The essence of our approach is encapsulated in the following lemma. Intuitively, it provides a sufficient condition for a cut in a given ordering π so that it can be assumed to be used in an optimum ordering σ.

▶ **Lemma 11.** *Let $D = (V, E)$ be a semi-complete digraph. Let π and σ be two vertex orderings of D such that $\mathsf{ctw}(D, \sigma) \leq \mathsf{ctw}(D, \pi) = c$. Suppose further that $m \in [4c, |V| - 4c]$ is such that in D there is a family of $|E^m_\pi|$ arc-disjoint paths leading from $\pi_{>m+4c}$ to $\pi_{\leq m-4c}$. Then there exists a vertex ordering σ^* such that:*

- $\sigma^*_{\leq m} = \pi_{\leq m}$;
- *for every j with $j \leq m - 4c$ or $j > m + 4c$, we have $\sigma^*_j = \sigma_j$;*
- $\mathsf{ctw}(D, \sigma^*) \leq \mathsf{ctw}(D, \sigma)$.

The intuition behind Lemma 11 is as follows. Consider σ^* as rearranged σ. The second condition says that this rearrangement is local: it affects only vertices at positions in the range $[m-4c+1, m+4c]$. The third condition says that the rearrangement does not increase the width. Finally, the first condition is crucial: σ^* uses the prefix $\pi_{\leq m}$ of π as one of its prefixes. Thus, any ordering can be locally rearranged while preserving the width so that prefix $\pi_{\leq m}$ is used, provided there is a large arc-disjoint flow locally near m.

Proof of Lemma 11. We first establish the following basic observation on the relation between orderings π and σ.

▶ **Claim 12 (♠).** *In the ordering σ, every vertex of $\pi_{\leq m-4c}$ is placed before every vertex of $\pi_{>m}$, and every vertex of $\pi_{\leq m}$ is placed before every vertex of $\pi_{>m+4c}$.*

The proof of the claim naturally follows by finding, say for each $u \in \pi_{\leq m-4c}$ and $v \in \pi_{>m}$, sufficiently many vertices that are both outneighbors of u and inneighbors of v.

Let $\sigma_\leq$ and $\sigma_>$ denote the restriction of σ to $\pi_{\leq m}$ and $\pi_{>m}$, respectively. Then, define σ^* to be the concatenation of $\sigma_\leq$ and $\sigma_>$. By the construction we have $\pi_{\leq m} = \sigma^*_{\leq m}$, so the first condition is satisfied. For the second condition, observe that by Claim 12, every vertex of $\pi_{\leq m-4c}$ is before every vertex of $\pi_{>m}$ in σ. It follows that in σ, the first vertex of $\pi_{>m}$ appears only after a prefix of at least $m - 4c$ vertices of $\pi_{\leq m}$. In the construction of σ^* from σ, the vertices of that prefix stay at their original positions, so $\sigma^*_j = \sigma_j$ for all $j \leq m - 4c$. A symmetric argument shows that $\sigma^*_j = \sigma_j$ also for all $j > m + 4c$.

It remains to prove that $\mathsf{ctw}(D, \sigma^*) \leq \mathsf{ctw}(D, \sigma)$. Consider any $j \in [0, |V|]$; we need to prove that $|E^j_{\sigma^*}| \leq \mathsf{ctw}(D, \sigma)$. By the second condition we have that $E^j_{\sigma^*} = E^j_\sigma$ when $j \leq m - 4c$ or $j \geq m + 4c$, and $|E^j_\sigma| \leq \mathsf{ctw}(D, \sigma)$ by definition. Hence, we are left with checking the inequality for j satisfying $m - 4c < j < m + 4c$.

In the following, for a vertex subset A we denote $\delta(A) = |E(V \setminus A, A)|$. We will use the submodularity of directed cuts: $\delta(A \cap B) + \delta(A \cup B) \leq \delta(A) + \delta(B)$ for all vertex subsets A, B. In these terms, we need to prove that $\delta(\sigma^*_{\leq j}) \leq \mathsf{ctw}(D, \sigma)$.

Let x be the vertex at position j in σ^* and let X be the set of all vertices placed not after x in σ, including x itself. Suppose first that $j \leq m$. Then, by the construction we have $x \in \pi_{\leq m}$ and $\sigma^*_{\leq j} = X \cap \pi_{\leq m}$. By the submodularity of cuts we have

$$\delta(\sigma^*_{\leq j}) = \delta(X \cap \pi_{\leq m}) \leq \delta(X) + \delta(\pi_{\leq m}) - \delta(X \cup \pi_{\leq m}). \tag{1}$$

As X is a prefix of σ by definition, we have $\delta(X) \leq \mathsf{ctw}(D, \sigma)$. Hence, by (1), in order to prove that $\delta(\sigma^*_{\leq j}) \leq \mathsf{ctw}(D, \sigma)$, it suffices to prove that $\delta(X \cup \pi_{\leq m}) \geq \delta(\pi_{\leq m})$.

Denote $d = \delta(\pi_{\leq m}) = |E^m_\pi|$ and recall that there is a family of d arc-disjoint paths leading from $\pi_{>m+4c}$ to $\pi_{\leq m-4c}$. In particular, this means that for each set A with $A \supseteq \pi_{\leq m-4c}$ and $A \cap \pi_{>m+4c} = \emptyset$, each of these paths has to contribute to $\delta(A)$, implying $\delta(A) \geq d$.

Therefore, it suffices to show that $X \cup \pi_{\leq m} \supseteq \pi_{\leq m-4c}$ and $(X \cup \pi_{\leq m}) \cap \pi_{>m+4c} = \emptyset$. While the first assertion is trivial, the second is equivalent to $X \cap \pi_{>m+4c} = \emptyset$. For this, observe that by definition all elements of X are placed not after x in σ, and x belongs to $\pi_{\leq m}$. However, by Claim 12 all vertices of $\pi_{>m+4c}$ are placed in σ after all vertices of $\pi_{\leq m}$, in particular after x. This implies that X and $\pi_{>m+4c}$ are disjoint, which proves that $\delta(X \cup \pi_{\leq m}) \geq d$ and, consequently as discussed above, also that $\delta(\sigma^*_{\leq j}) \leq \mathsf{ctw}(D, \sigma)$.

The proof for the case $j > m$ is completely symmetric, however we need to observe that now $x \in \pi_{>m}$ and $\sigma^*_{\leq j} = X \cup \pi_{\leq m}$. By applying the same submodularity argument (1), we are left with proving that $\delta(X \cap \pi_{\leq m}) \geq \delta(\pi_{\leq m})$, which follows by a symmetric reasoning. ◀

Our goal now is to construct an approximate ordering π where we will be able to find many positions m to which Lemma 11 can be applied. We first recall the concept of a lean ordering, which will be our main tool for finding families of arc-disjoint paths.

▶ **Definition 13.** A vertex ordering π of a digraph $D = (V, E)$ is called *lean* if for each $0 \leq a \leq b \leq n$, the maximum size of a family of arc-disjoint paths from $\pi_{>b}$ to $\pi_{\leq a}$ in D is equal to $\min_{a \leq i \leq b} |E^i_\pi|$.

Note that by Menger's theorem, the maximum size of a family of arc-disjoint paths from $\pi_{>b}$ to $\pi_{\leq a}$ is equal to the minimum size of an arc cut separating $\pi_{>b}$ from $\pi_{\leq a}$. Thus, in a lean ordering we have that the minimum cutsize between any disjoint prefix and suffix is actually realized by one of the cuts along the ordering.

The notion of a lean ordering is the cutwidth analogue of a *lean decomposition* in the treewidth setting, cf. [20]. An essentially equivalent notion of *linked orderings* was used by Chudnovsky and Seymour [4] in the context of immersions in tournaments. Also, Giannopoulou et al. [12] used this concept to study immersion obstructions for the cutwidth of undirected graphs. A careful analysis of the arguments of [4, 12] yields the following.

▶ **Lemma 14** ([4, 12])**.** *There is a polynomial-time algorithm that given a vertex ordering π of a digraph D, computes a lean vertex ordering π^* of D satisfying $\mathsf{ctw}(D, \pi^*) \leq \mathsf{ctw}(D, \pi)$.*

Next, we introduce the concept of a *milestone*. Intuitively, a milestone is a position where Lemma 11 can be applied, provided the ordering is lean.

▶ **Definition 15.** Let π be a vertex ordering of a digraph $D = (V, E)$, and let α be a positive integer. An integer $m \in [0, |V|]$ is a *π-milestone of D of span α* if $|E^m_\pi| \leq |E^i_\pi|$ for each integer i with $m - \alpha \leq i \leq m + \alpha$.

Note that if π is lean and m is a π-milestone of span α, then $\min_{m-\alpha \leq i \leq m+\alpha} |E^i_\pi| = |E^m_\pi|$, hence there is a family of $|E^\pi_m|$ arc-disjoint paths leading from $\pi_{>m+\alpha}$ to $\pi_{\leq m-\alpha}$. Thus, a π-milestone of span $4c$ satisfies the prerequisite of Lemma 11 about the existence of arc-disjoint paths. We now observe that in an ordering of small width milestones occur often.

▶ **Lemma 16 (♠).** *Let $D = (V, E)$ be a digraph and let π be a vertex ordering of D of width at most c. Then for any integers $p \in [0, |V|]$ and $\alpha \geq 0$, there exists a π-milestone $m \in [p - \alpha \cdot c, p + \alpha \cdot c]$ of span α.*

Having gathered all the tools, we are finally ready to prove Theorem 4.

Proof of Theorem 4. By Theorem 8, we can compute in polynomial time a vertex ordering π_0 of D such that $\mathsf{ctw}(D, \pi_0) \leq 2 \cdot \mathsf{ctw}(D)$. If $\mathsf{ctw}(D, \pi_0) > 2c$, we can conclude that $\mathsf{ctw}(D) > c$ and report this answer, so let us assume that $\mathsf{ctw}(D, \pi_0) \leq 2c$. By applying the algorithm of Lemma 14 to π_0, we can compute in polynomial time a lean ordering π such that $\mathsf{ctw}(D, \pi) \leq \mathsf{ctw}(D, \pi_0) \leq 2c$. In the following we assume w.l.o.g. that $|V| > 16c$, for otherwise we can output a list consisting only of D.

Call a set of π-milestones *dispersed* if these π-milestone pairwise differ by more than $16c$. Observe that 0 and $|V|$ are always π-milestones, and they differ by more than $16c$. Starting from the set $\{0, |V|\}$, we compute an inclusion-wise maximal dispersed set $0 = m_0 < m_1 < m_2 < \ldots < m_\ell = |V|$ of π-milestones of span $8c$. More precisely, whenever some π-milestone of span $8c$ can be added to the set without spoiling the dispersity requirement, we do it, until no further such milestone can be added. Observe that then we have that $m_{i+1} - m_i \leq 32c^2 + 32c + 1$ for each $i \in [1, \ell - 1]$, for otherwise the range $[m_i + 16c + 1, m_{i+1} - 16c - 1]$ would contain more than $32c^2$ vertices, so by Lemma 16 we would be able to find in it a π-milestone of span $8c$ that could be added to the constructed dispersed set.

Thus, π is partitioned into ℓ blocks $B_1, \ldots, B_\ell$, each of length at most $32c^2 + 32c + 1$, such that the j-th block B_j is equal to $\{\pi_{m_{j-1}+1}, \pi_{m_{j-1}+2}, \ldots, \pi_{m_j}\}$. For each $j \in [1, \ell]$, let A_j be defined as B_j augmented with the following vertices:

- vertices at positions in ranges $[\max(1, m_{j-1} - 8c + 1), m_{j-1}]$ and $[m_j + 1, \min(|V|, m_j + 8c)]$,
- all heads of arcs from $E_\pi^{m_{j-1}-8c}$, and all tails of arcs from $E_\pi^{m_j+8c}$.

Since the width of π is at most $2c$, we have that $|A_j| \leq |B_j| + 20c = \mathcal{O}(c^2)$.

For $j \in [1, \ell]$, let us denote $D_j = D[A_j]$. To prove the theorem, it now suffices to show that $\mathsf{ctw}(D) \leq c$ if and only if $\mathsf{ctw}(D_j) \leq c$ for each $j \in [1, \ell]$. The forward direction is trivial, since cutwidth is closed under taking induced subdigraphs. Hence, we are left with showing that if $\mathsf{ctw}(D_j) \leq c$ for each $j \in [1, \ell]$, then $\mathsf{ctw}(D) \leq c$.

Take any $j \in [1, \ell - 1]$. As m_j is a π-milestone of span $8c$, we have $\min_{m_j - 8c \leq i \leq m_j + 8c} |E_\pi^i| = |E_\pi^{m_j}|$. Since π is lean, there is a family $\mathcal{F}_j$ of $|E_\pi^{m_j}|$ arc-disjoint paths in D leading from $\pi_{>m_j+8c}$ to $\pi_{\leq m_j-8c}$. We can assume w.l.o.g. that each internal (non-endpoint) vertex of each of these paths has position between $m_j + 8c + 1$ and $m_j - 8c$ in π. Hence, in particular, each path of $\mathcal{F}_j$ starts with an arc of $E_\pi^{m_j+8c}$ and ends with an arc of $E_\pi^{m_j-8c}$. This implies that for each $j \in [1, \ell]$, all the paths of $\mathcal{F}_j$ are entirely contained both in D_j and in D_{j+1}.

Consider any $j \in [1, \ell]$, and for simplicity assume for now that $j \neq 1$ and $j \neq \ell$. Let π' be the restriction of π to the vertex set of D_j; obviously the width of π' is at most $2c$. Further, let m' be the position of $\pi_{m_{j-1}}$ in π', so that $\pi'_{\leq m'} = \pi_{\leq m_{j-1}} \cap V(D_j)$. Observe that since all vertices at positions between $m_{j-1} - 8c + 1$ and $m_{j-1} + 8c$ in π are included in the vertex set of D_j, they are at positions between $m' - 8c + 1$ and $m' + 8c$ in π', and hence the paths of $\mathcal{F}_{j-1}$ in D_j lead from $\pi'_{>m'+8c}$ to $\pi'_{\leq m'-8c}$. Their number is $|E_\pi^{m_{j-1}}|$, which is equal to the cutsize at position m' in π', by the construction of D_j and π'.

We conclude that Lemma 11 can be applied to position m' in the ordering π' of D_j. If we now use it on any ctw-optimal vertex ordering σ of D_j, we obtain a ctw-optimal vertex ordering σ^* of D_j such that $\sigma^*_{\leq m'} = \pi'_{\leq m'} = \pi_{\leq m_{j-1}} \cap V(D_j)$. Note that by Lemma 11, σ^* differs from σ by a rearrangement of vertices at positions between $m' - 8c + 1$ and $m' + 8c$.

Now we define m'' to be the position of π_{m_j} in π', so that $\pi'_{\leq m''} = \pi_{\leq m_j} \cap V(D_j)$. A symmetric reasoning, which uses the fact that $\mathcal{F}_j$ is also entirely contained in D_j, shows that Lemma 11 can be also applied to position m'' in the ordering π' of D_j. Then we can use this lemma on the ctw-optimal vertex ordering σ^*, yielding a ctw-optimal ordering σ^{**} such that $\sigma^{**}_{\leq m''} = \pi'_{\leq m''} = \pi_{\leq m_j} \cap V(D_j)$. Again, by Lemma 11 we have that σ^* and σ^{**} differ by a rearrangement of vertices at positions $m'' - 8c + 1$ and $m'' + 8c$. Since $m_j - m_{j-1} > 16c$ by construction, we infer that this rearrangement does not change the prefix of length m', and hence we still have $\sigma^{**}_{\leq m'} = \pi'_{\leq m'} = \pi_{\leq m_{j-1}} \cap V(D_j)$. The ordering σ^{**} obtained in this manner shall be called σ^j. For $j = 1$ and $j = \ell$ we obtain σ^j in exactly the same way, except we apply Lemma 11 only once, for the position not placed at the end of the sequence.

All in all, for each $j \in [1, \ell]$ we have obtained a ctw-optimal ordering σ^j of D_j such that the vertices of B_j form an infix (a sequence of consecutive elements) of σ^j, while vertices to the left of this infix are the vertices of $V(D_j) \cap \pi_{\leq m_{j-1}}$ and vertices to the right of this infix are the vertices of $V(D_j) \cap \pi_{>m_j}$. Define an ordering σ of D by first restricting every ordering σ^j to B_j, and then concatenating all the obtained orderings for $j = 1, 2, \ldots, \ell$. Since we assumed that $\mathsf{ctw}(D_j) \leq c$ for each $j \in [1, \ell]$, and each ordering σ^j is ctw-optimal on D_j, we have that $\mathsf{ctw}(D_j, \sigma^j) \leq c$ for each $j \in [1, \ell]$. From the construction of D_j, and in particular the fact that all the arcs of $E_\pi^{m_{j-1}}$ and $E_\pi^{m_j}$ are contained in D_j, it follows that the infix of cutvector $\mathsf{cuts}\langle D^j, \sigma^j \rangle$ corresponding to the vertices of B_j is equal to the infix of the cutvector $\mathsf{cuts}\langle D, \sigma \rangle$ corresponding to the vertices of B_j. This shows that

$$\mathsf{ctw}(D, \sigma) = \max_{i \in [0, |V|]} \mathsf{cuts}\langle D, \sigma \rangle(i) \leq \max_{\substack{j \in [0,\ell] \\ i \in [0, |V(D_j)|]}} \mathsf{cuts}\langle D^j, \sigma^j \rangle(i) = \max_{j \in [0,\ell]} \mathsf{ctw}(D_j, \sigma^j) \leq c,$$

hence we are done. ◀

Regarding the bounds on sizes of c-cutwidth-minimal semi-complete digraphs (Theorems 5 and 6), we will give a full exposition in the complete version of the paper. Essentially, Theorem 5 follows easily by considering applying the algorithm of Theorem 4 on a c-cutwidth-minimal semi-complete digraph for parameter $c - 1$, while for Theorem 6 we need to use the understanding of minimum orderings in tournaments in the spirit of Lemma 10.

4 Lower bounds

In this section, we prove Theorem 1, which provides almost tight lower bounds for the complexity of computing the cutwidth and the Ola-cost of a semi-complete digraph. We start our reduction from an instance of the NAE-3SAT problem, which was defined in Section 2 and for which a complexity lower bound under ETH is given by Corollary 9.

Let us introduce some notation. For a formula φ in CNF, the variable and clause sets of φ are denoted by $\mathrm{vars}(\varphi)$ and $\mathrm{cls}(\varphi)$, respectively. A variable assignment $\alpha \colon \mathrm{vars}(\varphi) \to \{\bot, \top\}$ *NAE-satisfies* φ if every clause of φ has at least one, but not all literals satisfied. Formula φ is *NAE-satisfiable* if there is a variable assignment α that NAE-satisfies it; equivalently, both α and its negation $\neg\alpha$ satisfy φ. A digraph is called *basic* if it is simple and has no pair of symmetric arcs. For an integer $m > 0$, let λ_m be the tuple of size $14m + 1$ such that:

$$\lambda_m(i) = \begin{cases} 2i, & \text{when } i \in [0, 5m] \\ 5m + i, & \text{when } i \in [5m + 1, 6m] \\ 11m, & \text{when } i \in [6m + 1, 7m] \\ 18m - i, & \text{when } i \in [7m + 1, 12m] \\ 42m - 3i, & \text{when } i \in [12m + 1, 14m] \end{cases}$$

The following lemma encapsulates the first, main step of our reduction.

▶ **Lemma 17.** *There exists a polynomial-time algorithm that, given a 3CNF formula φ with m clauses, returns a basic digraph $D(\varphi)$ with $14m$ vertices and $24m$ edges such that:*

1. *for every vertex ordering π, we have $\mathsf{cuts}\langle D(\varphi), \pi\rangle \preceq \lambda_m$;*
2. *if φ is NAE-satisfiable, then there exists a vertex ordering π with $\mathsf{cuts}\langle D(\varphi), \pi\rangle = \lambda_m$;*
3. *if there is a vertex ordering π with $\max\{\mathsf{cuts}\langle D(\varphi), \pi\rangle\} \geq 11m$, then φ is NAE-satisfiable.*

Note that Lemma 17 expresses a reduction from NAE-3SAT to a maximization problem: NAE-satisfiability of φ is equivalent to $D(\varphi)$ admitting a vertex ordering of width *at least* $11m$. The main idea is that this maximization will be later turned into minimization by complementing the digraph, which also yields a semi-complete digraph since $D(\varphi)$ is basic.

Proof of Lemma 17. Without loss of generality, we may assume that each clause of φ contains exactly 3 literals, by repeating some literal if necessary. Then, we may also assume that every variable of $\mathrm{vars}(\varphi)$ appears at least twice, because a variable that appears only once can always be set in order that the clause in which it appears is NAE-satisfied, and thus such a variable and its associated clause may be safely removed. For every variable $x \in \mathrm{vars}(\varphi)$, let p_x be the number of occurrences of x in the clauses of φ; hence $3m = \sum_{x \in \mathrm{vars}(\varphi)} p_x$ and $p_x \geq 2$ for each $x \in \mathrm{vars}(\varphi)$. We finally assume the clauses and literals are ordered, so we may say that a literal ℓ_x is the i_xth occurrence of variable x in the clauses of φ, with $i_x \in [1, p_x]$.

We now describe the construction of $D(\varphi)$. For every variable $x \in \mathrm{vars}(\varphi)$ construct a *variable gadget* G_x, which is a directed cycle of length $2p_x$ with vertices named as follows:

$$\bot^x_1 \to \top^x_1 \to \bot^x_2 \to \top^x_2 \to \ldots \to \bot^x_{p_x} \to \top^x_{p_x} \to \bot^x_1.$$

Note that this cycle has no symmetric arcs since $p_x > 1$.

Then, for every clause $C \in \mathrm{cls}(\varphi)$, where $C = \ell_x \vee \ell_y \vee \ell_z$ for literals of variables $x, y, z \in \mathrm{vars}(\varphi)$, respectively, construct the following *$\top$-clause gadget* $G^C_\top$. Introduce a vertex $\top^C$ and a set of vertices $V^C_\top = \{\top^C_{\ell_x}, \top^C_{\ell_y}, \top^C_{\ell_z}\}$ together with the following arcs:

- A directed 3-cycle $(\top^C_{\ell_x}, \top^C_{\ell_y}), (\top^C_{\ell_y}, \top^C_{\ell_z}), (\top^C_{\ell_z}, \top^C_{\ell_x})$.
- The arcs $(\top^C, \top^C_{\ell_x}), (\top^C, \top^C_{\ell_y})$ and $(\top^C, \top^C_{\ell_z})$ from $\top^C$ to the vertices of V^C_T.

Similarly, construct the *$\bot$-clause gadget* $G^C_\bot$, which is isomorphic to $G^C_\top$, but with vertices named $\bot$. Gadgets $G^C_\top$ and $G^C_\bot$ will differ in how we connect them with the rest of the graph.

Intuitively, the variable assignment α, intended to NAE-satisfy φ, is encoded by choosing, in each variable gadget G_x, which vertices are placed in the first half of π, and which are placed in the second. We use the gadget $G^C_\top$ to verify that α satisfies C, whereas the gadget $G^C_\bot$ verifies that $\neg\alpha$ also satisfies C. For this purpose, connect the clause gadgets to variable gadgets as follows. Suppose $\ell_x \in C$ is the i_xth occurrence of x. If $\ell_x = x$ then add two arcs $(\top^C_{\ell_x}, \bot^x_{i_x})$ and $(\bot^C_{\ell_x}, \top^x_{i_x})$, and if $\ell_x = \neg x$ then add two arcs $(\top^C_{\ell_x}, \top^x_{i_x})$ and $(\bot^C_{\ell_x}, \bot^x_{i_x})$.

This concludes the construction of $D(\varphi)$. Clearly $D(\varphi)$ is basic, and a straightforward verification using the equality $3m = \sum_{x \in \mathrm{vars}(\varphi)} p_x$ shows that conditions $|V(D(\varphi))| = 14m$ and $|E(D(\varphi))| = 24m$ hold as well. The complete proof of the three lemma statements will appear in the full version of the paper. In order to show the main gist of the reduction, we sketch now the proof of the third claim.

We prove the following statement: for any vertex subset $A \subseteq V$, it always holds that $|E(A, V \setminus A)| \leq 11m$. Note that this in particular implies that the cutwidth of any ordering of $D(\varphi)$ is at most $11m$, which is a part of the verification of the first claim. Denote $F = E(A, V \setminus A)$. First, consider any variable $x \in \mathrm{vars}(\varphi)$. Since G_x is a directed cycle of length $2p_x$, it can easily be seen that $|F \cap E(G_x)| \leq p_x$ and the equality holds if and only if A contains every second vertex of the cycle G_x. Second, consider any clause $C = \ell_x \vee \ell_y \vee \ell_z \in \mathrm{cls}(\varphi)$. Let $R^C_\top$ be the set of three arcs connecting G^C with the variable

gadgets G_x, G_y, and G_z. Since $\top^C$ has no incoming arcs, we can assume without loss of generality that $\top^C \in A$, as putting $\top^C$ into A can only increase $|E(A, V \setminus A)|$. We now distinguish cases depending on the size of $A \cap V_\top^C = A \cap \{\top_{\ell_x}^C, \top_{\ell_y}^C, \top_{\ell_z}^C\}$. The following implications follow from a straightforward analysis of the situation in $G_\top^C$ and on incident arcs.

- If $|A \cap V_\top^C| = 0$ then $|F \cap R_\top^C| = 0$ and $|F \cap E(G_\top^C)| = 3$.
- If $|A \cap V_\top^C| = 1$ then $|F \cap R_\top^C| \leq 1$ and $|F \cap E(G_\top^C)| = 3$.
- If $|A \cap V_\top^C| = 2$ then $|F \cap R_\top^C| \leq 2$ and $|F \cap E(G_\top^C)| = 2$.
- If $|A \cap V_\top^C| = 3$ then $|F \cap R_\top^C| \leq 3$ and $|F \cap E(G_\top^C)| = 0$.

In all the cases, we conclude that $|F \cap (E(G_\top^C) \cup R_\top^C)| \leq 4$; note that the equality can hold only in the two middle ones. The same analysis applies to the $\bot$-clause gadgets, yielding $|F \cap (E(G_\bot^C) \cup R_\bot^C)| \leq 4$, where $R_\bot^C$ is defined analogously. Since the sets $E(G_x)$ for $x \in \mathrm{vars}(\varphi)$ and $E(G_\top^C) \cup R_\top^C \cup E(G_\bot^C) \cup R_\bot^C$ for $C \in \mathrm{cls}(\varphi)$ form a partition of $E(D(\varphi))$, we immediately get that $|F| \leq \sum_{x \in \mathrm{vars}(\varphi)} p_x + 8|\mathrm{cls}(\varphi)| = 11m$.

To verify the third claim of the lemma, note that if there is some vertex ordering of cutwidth at least $11m$, then there is some set A with $|E(A, V \setminus A)| \geq 11m$. Hence, for such all the inequalities used above are in fact equalities. In particular, in every variable gadget G_x, the vertices belong to A and to $V(D) \setminus A$ alternately. This gives us two possibilities for every variable gadget, which naturally defines a variable assignment α for the formula φ. The fact that we have equalities also in each clause gadget $G_\top^C$ and $G_\bot^C$ ensures that each clause C is satisfied both by α and $\neg\alpha$. Hence α NAE-satisfies φ. ◀

We now proceed to complementing the obtained digraph. Precisely, given a simple digraph $D = (V, E)$, define its *complement* as $\bar{D} = (V, \bar{E})$, where $\bar{E} = V^2 \setminus (E \cup \{(u, u) \colon u \in V\})$. That is, we take the complete digraph without self-loops on the vertex set V, and we remove all the arcs that are present in D. Note that the complement of a basic digraph is semi-complete.

Now, let $\bar{\lambda}_m$ be the tuple such that for all $i \in [0, 14m]$, we have $\lambda_m(i) + \bar{\lambda}_m(i) = i(14m - i)$. It is not hard to check that $\max\{\bar{\lambda}_m\} = \bar{\lambda}_m(7m) = 49m^2 - 11m$. A simple verification of how the conditions of Lemma 17 are transformed under complementation yields the following.

▶ **Lemma 18 (♠).** *The complement of $D(\varphi)$ is a semi-complete digraph $\bar{D}(\varphi)$ satisfying:*

1. *for every vertex ordering π, we have $\bar{\lambda}_m \preceq \mathsf{cuts}\langle \bar{D}(\varphi), \pi \rangle$;*
2. *if φ is NAE-satisfiable, then there exists a vertex ordering π with $\mathsf{cuts}\langle \bar{D}(\varphi), \pi \rangle = \bar{\lambda}_m$;*
3. *if $\bar{D}(\varphi)$ admits a vertex ordering π of width at most $49m^2 - 11m$, then φ is NAE-satisfiable.*

Thus, Lemma 18 shows that NAE-satisfiability of φ is equivalent to $\bar{D}(\varphi)$ having cutwidth at most $49m^2 - 11m$. However, the fact that NAE-satisfiability of φ implies that $\bar{D}(\varphi)$ admits a vertex ordering with a very concrete cut vector $\bar{\lambda}_m$, which is the best possible in the sense of the first claim of Lemma 18, also enables us to derive a lower bound for OLA. All these observations, together with the linear bound on the number of vertices of $\bar{D}(\varphi)$, make the proof of Theorem 1 essentially complete.

The reduction of Lemma 17 constructs a basic digraph whose complement has a pair of symmetric arcs between almost every pair of vertices. On the other hand, on tournaments the problem is polynomial-time solvable, which suggests looking at the parameterization by the number of vertices incident to symmetric arcs. We indeed show that this parameterization leads to an FPT problem, even in a larger generality. Call a vertex u of a simple digraph D *pure* if for any other vertex v, exactly one of the arcs (u, v) or (v, u) is present in D.

▶ **Theorem 19 (♠).** *There is an algorithm that, given a simple digraph D on n vertices, computes the cutwidth and the OLA-cost of D in time $2^k \cdot n^{\mathcal{O}(1)}$, where k is the number of non-pure vertices in D. The algorithm can also report orderings certifying the output values.*

References

1 Jørgen Bang-Jensen and Gregory Gutin. *Digraphs – theory, algorithms and applications*. Springer, 2002.

2 Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, 8(4):38, 2012.

3 Maria Chudnovsky, Alexandra Fradkin, and Paul Seymour. Tournament immersion and cutwidth. *Journal of Combinatorial Theory, Series B*, 102(1):93–101, 2012.

4 Maria Chudnovsky and Paul Seymour. A well-quasi-order for tournaments. *Journal of Combinatorial Theory, Series B*, 101(1):47–53, 2011.

5 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

6 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

7 Andrew Drucker. New limits to classical and quantum instance compression. *SIAM journal of Computing*, 44(5):1443–1479, 2015.

8 Fedor V. Fomin and Michał Pilipczuk. Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph. In *ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 505–516. Springer, 2013.

9 Alexandra Fradkin. *Forbidden structures and algorithms in graphs and digraphs*. PhD thesis, Princeton University, 2011.

10 Alexandra Ovetsky Fradkin and Paul D. Seymour. Edge-disjoint paths in digraphs with bounded independence number. *Journal of Combinatorial Theory, Series B*, 110:19–46, 2015.

11 Valentin Garnero and Mathias Weller. Parameterized certificate dispersal and its variants. *Theoretical Computer Science*, 622:66–78, 2016.

12 Archontia C. Giannopoulou, Michał Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: obstructions and algorithmic aspects. *CoRR*, abs/1606.05975, 2016. To appear in Proceedings of IPEC 2016.

13 Danny Hermelin, Stefan Kratsch, Karolina Sołtys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (Turing) kernelization. *Algorithmica*, 71(3):702–730, 2015.

14 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

15 Bart M. P. Jansen. Turing kernelization for finding long paths and cycles in restricted graph classes. *Journal of Computer and System Sciences*, 85:18–37, 2017.

16 Michał Pilipczuk. Computing cutwidth and pathwidth of semi-complete digraphs via degree orderings. In *STACS 2013*, volume 20 of *LIPIcs*, pages 197–208. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013.

17 Michał Pilipczuk. *Tournaments and optimality: new results in parameterized complexity*. PhD thesis, University of Bergen, Norway, 2013.

18 Alexander Schäfer, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012.

19 Thomas J. Shaefer. The complexity of satisfiability problems. In *STOC 1978*, pages 216–226. ACM, 1978.

20 Robin Thomas. A menger-like property of tree-width: the finite case. *Journal of Combinatorial Theory Series B*, 48(1):67–76, 1990.

21 Stéphan Thomassé, Nicolas Trotignon, and Kristina Vušković. A polynomial Turing-kernel for Weighted Independent Set in bull-free graphs. *Algorithmica*, 77(3):619–641, 2017.

Packing Cycles Faster Than Erdős-Pósa*

Daniel Lokshtanov[1], Amer E. Mouawad[1], Saket Saurabh[3], and Meirav Zehavi[1]

1 **University of Bergen, Bergen, Norway**
daniel.lokshtanov@ii.uib.no
2 **University of Bergen, Bergen, Norway**
a.mouawad@ii.uib.no
3 **University of Bergen, Bergen, Norway; and**
The Institute of Mathematical Sciences, Chennai, India
saket@imsc.res.in
4 **University of Bergen, Bergen, Norway**
meirav.zehavi@ii.uib.no

Abstract

The Cycle Packing problem asks whether a given undirected graph $G = (V, E)$ contains k vertex-disjoint cycles. Since the publication of the classic Erdős-Pósa theorem in 1965, this problem received significant scientific attention in the fields of Graph Theory and Algorithm Design. In particular, this problem is one of the first problems studied in the framework of Parameterized Complexity. The non-uniform fixed-parameter tractability of Cycle Packing follows from the Robertson–Seymour theorem, a fact already observed by Fellows and Langston in the 1980s. In 1994, Bodlaender showed that Cycle Packing can be solved in time $2^{\mathcal{O}(k^2)} \cdot |V|$ using exponential space. In case a solution exists, Bodlaender's algorithm also outputs a solution (in the same time). It has later become common knowledge that Cycle Packing admits a $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time (deterministic) algorithm using exponential space, which is a consequence of the Erdős-Pósa theorem. Nowadays, the design of this algorithm is given as an exercise in textbooks on Parameterized Complexity. Yet, no algorithm that runs in time $2^{o(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, has been found. In light of this, it seems natural to ask whether the $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$ bound is essentially optimal. In this paper, we answer this question negatively by developing a $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$-time (deterministic) algorithm for Cycle Packing. In case a solution exists, our algorithm also outputs a solution (in the same time). Moreover, apart from beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, our algorithm runs in time linear in $|V|$, and its space complexity is polynomial in the input size.

1998 ACM Subject Classification G.2.2 Graph Algorithms, I.1.2 Analysis of Algorithms

Keywords and phrases Parameterized Complexity, Graph Algorithms, Cycle Packing, Erdős-Pósa Theorem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.71

1 Introduction

The Cycle Packing problem asks whether a given undirected graph $G = (V, E)$ contains k vertex-disjoint cycles. Since the publication of the classic Erdős-Pósa theorem in 1965 [15], this problem received significant scientific attention in the fields of Graph Theory and Algorithm Design. In particular, Cycle Packing is one of the first problems studied in the framework of Parameterized Complexity. In this framework, each problem instance is

* A full version of the paper is available at https://arxiv.org/abs/1707.01037.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 71; pp. 71:1–71:15

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

associated with a parameter k that is a non-negative integer, and a problem is said to be *fixed-parameter tractable (FPT)* if the combinatorial explosion in the time complexity can be confined to the parameter k. More precisely, a problem is FPT if it can be solved in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some function f, where $|I|$ is the input size. For more information, we refer the reader to recent monographs such as [14] and [10].

In this paper, we study the Cycle Packing problem from the perspective of Parameterized Complexity. In the standard parameterization of Cycle Packing, the parameter is the number k of vertex-disjoint cycles. The non-uniform fixed-parameter tractability of Cycle Packing follows from the Robertson–Seymour theorem [38],[1] a fact already observed by Fellows and Langston in the 1980s. In 1994, Bodlaender showed that Cycle Packing can be solved in time $2^{\mathcal{O}(k^2)} \cdot |V|$ using exponential space [5]. Notably, in case a solution exists, Bodlaender's algorithm also outputs a solution in time $2^{\mathcal{O}(k^2)} \cdot |V|$.

The Erdős-Pósa theorem states that there exists a function $f(k) = \mathcal{O}(k \log k)$ such that for each non-negative integer k, every undirected graph either contains k vertex-disjoint cycles or it has a feedback vertex set consisting of $f(k)$ vertices [15]. It is well known that the treewidth (tw) of a graph is not larger than its feedback vertex set number (fvs), and that a naive dynamic programming (DP) scheme solves Cycle Packing in time $2^{\mathcal{O}(tw \log tw)} \cdot |V|$ and exponential space (see, e.g., [10]). Thus, the existence of a $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time (deterministic) algorithm that uses exponential space can be viewed as a direct consequence of the Erdős-Pósa theorem. Nowadays, the design of this algorithm is given as an exercise in textbooks on Parameterized Complexity such as [14] and [10]. In case a solution exists, this algorithm does not output a solution (though we remark that with a certain amount of somewhat non-trivial work, it is possible to modify this algorithm to also output a solution).

Prior to our work, no algorithm that runs in time $2^{o(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, has been found. In light of this, it seemed tempting to ask whether the $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$ bound is essentially optimal. In particular, two natural directions to explore in order to obtain a faster algorithm necessarily lead to a dead end. First, Erdős and Pósa [15] proved that the bound $f(k) = \mathcal{O}(k \log k)$ in their theorem is essentially tight as there exist infinitely many graphs and a constant c such that these graphs do not contain k vertex-disjoint cycles and yet their feedback vertex set number is larger than $ck \log k$. Second, Cygan et al. [11] proved that the bound $2^{\mathcal{O}(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ is also likely to be essentially tight in the sense that unless the Exponential Time Hypothesis (ETH) [20] is false, Cycle Packing cannot be solved in time $2^{o(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ (however, it might still be true that Cycle Packing is solvable in time $2^{o(fvs \log fvs)} \cdot |V|^{\mathcal{O}(1)}$).

1.1 Related Work

The Cycle Packing problem admits a factor $\mathcal{O}(\log |V|)$ approximation algorithm [30], and it is quasi-NP-hard to approximate within a factor of $\mathcal{O}(\log^{\frac{1}{2}-\epsilon} |V|)$ for any $\epsilon > 0$ [18]. In the context of kernelization with respect to the parameter k, Cycle Packing does not admit a polynomial kernel unless NP $\subseteq$ coNP/Poly [4]. Recently, Lokshtanov et al. [31] obtained a 6-approximate kernel with $\mathcal{O}((k \log k)^2)$ vertices along with a $(1+\epsilon)$-approximate kernel with $k^{\mathcal{O}(f(\epsilon))}$ vertices for some function f. We would like to mention that in case one seeks k edge-disjoint cycles rather than k vertex-disjoint cycles, the problem becomes significantly simpler in the sense that it admits a kernel with $\mathcal{O}(k \log k)$ vertices [4].

Focusing on structural parameters, Bodlaender et al. [2] obtained polynomial kernels with respect to the vertex cover number, vertex-deletion distance to a cluster graph and

[1] The paper [38] was already available as a manuscript in 1986 (see, e.g., [5]).

the max leaf number. In planar graphs, Bodlaender et al. [3] solved Cycle Packing in subexponential time $2^{\mathcal{O}(\sqrt{k})} \cdot |V|^{\mathcal{O}(1)}$, and showed that this problem admits a linear kernel. In the more general class of H-minor-free graphs, Dorn et at. [13] also solved Cycle Packing in subexponential time $2^{\mathcal{O}(\sqrt{k})} \cdot |V|^{\mathcal{O}(1)}$. Moreover, for apex-minor-free graphs, Fomin et al. [17] showed that Cycle Packing admits a linear kernel, and Fomin et al. [16] showed that it also admits an EPTAS. When the input graph is a directed graph, Cycle Packing is W[1]-hard [39], but it admits an FPT approximation scheme [19]. In fact, Cycle Packing in directed graphs was the first W[1]-hard problem shown to admit such a scheme. Krivelevich et al. [30] obtained a factor $\mathcal{O}(|V|^{\frac{1}{2}})$ approximation algorithm for Cycle Packing in directed graphs and showed that this problem is quasi-NP-hard to approximate within a factor of $\mathcal{O}(\log^{1-\epsilon} |V|)$ for any $\epsilon > 0$.

Several variants of Cycle Packing have also received significant scientific attention. For example, the variant of Cycle Packing where one seeks k *odd* vertex-disjoint cycles has been widely studied [36, 40, 35, 29, 27, 28]. Another well-known variant, where the cycles need to contain a prescribed set of vertices, has also been extensively investigated [24, 33, 25, 23, 26]. Furthermore, a combination of these two variants has been considered in [23, 22].

Finally, we briefly mention that inspired by the Erdős-Pósa theorem, a class of graphs $\mathcal{H}$ is said to have the Erdős-Pósa property if there is a function $f(k)$ for which given a graph G, it either contains k vertex-disjoint subgraphs such that each of these subgraphs is isomorphic to a graph in $\mathcal{H}$, or it contains a set of $f(k)$ vertices that hits each of its subgraphs that is isomorphic to a graph in $\mathcal{H}$. A fundamental result in Graph Theory by Robertson and Seymour [37] states the the class of all graphs that can be contracted to a fixed planar graph H has the Erdős-Pósa property. Recently, Chekuri and Chuzhoy [6] presented a framework that leads to substantially improved functions $f(k)$ in the context of results in the spirit of the Erdős-Pósa theorem. Among other results, these two works are also related to the recent breakthrough result by Chekuri and Chuzhoy [7], which states that every graph of treewidth at least $f(k) = \mathcal{O}(k^{98} \cdot \text{polylog}(k))$ contains the $k \times k$ grid as a minor (the constant 98 has been improved to 36 in [8] and to 19 in [9]). Following the seminal work by Robertson and Seymour [37], numerous papers (whose survey is beyond the scope of this paper) investigated which other classes of graphs have the Erdős-Pósa property, which are the "correct" functions f associated with them, and which generalizations of this property lead to interesting discoveries.

1.2 Our Contribution

In this paper, we show that the running time of the algorithm that is a consequence of the Erdős-Pósa theorem is not essentially tight. For this purpose, we develop a $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$-time (deterministic) algorithm for Cycle Packing. In case a solution exists, our algorithm also outputs a solution (in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$). Moreover, apart from beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, our algorithm runs in time linear in $|V|$, and its space complexity is polynomial in the input size. Thus, we also improve upon the classical $2^{\mathcal{O}(k^2)} \cdot |V|$-time algorithm by Bodlaender [5]. Our result is summarized in the following theorem.

▶ **Theorem 1.** *There exists a (deterministic) polynomial-space algorithm that solves* Cycle Packing *in time* $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$. *In case a solution exists, it also outputs a solution.*

At a high level to prove Theorem 1, the main idea is to look back at the classic algorithm based on the Erdős-Pósa property and to perform some trade-offs to get an improvement. Specifically, the idea is to compute a "relaxed feedback vertex set": a set S of size

$\mathcal{O}(k \log k / \log\log k)$ so that $G - S$ is not necessarily a forest, but it has no short cycles (of length $\mathcal{O}(\log k / \log\log k)$), even after contracting long detached paths to single edges. This creates a "relaxed" modulator that is smaller by a $\mathcal{O}(\log\log k)$ multiplicative factor than if we required S to be a feedback vertex set. After some cleaning step on this structure, and using the relaxed modulator, one can roughly guess the interaction between S and $G - S$ in the solution; this step amounts to guessing one of around $|S|!$ options, which is $\mathcal{O}(2^{k \log^2 k / \log\log k})$. Having fixed the interaction, we can contract any remaining long paths so that the whole remaining graph has $\mathcal{O}(k \log^{3/2} k)$ vertices; it is important here that this step goes through even when we work with the "relaxed" modulator S as explained above, instead of a normal feedback vertex set as in the classic algorithm. As the size of the graph is already bounded, a simple dynamic programming on subsets suffices to finish the proof. In order to achieve polynomial space usage, we employ a standard inclusion-exclusion procedure instead.

2 Preliminaries

We use standard terminology from the book of Diestel [12] for those graph-related terms that are not explicitly defined here. We only consider finite graphs possibly having self-loops and multi-edges. Moreover, using an appropriate reduction rule we restrict the maximum multiplicity of an edge to be 2. For a graph G, we use V and E to denote the vertex and edge sets of the graph G, respectively. For a vertex $v \in V$, we use $\deg_G(v)$ to denote the degree of v, i.e the number of edges incident on v, in the (multi) graph G. We also use the convention that a self-loop at a vertex v contributes 2 to its degree. For a vertex subset $S \subseteq V$, we let $G[S]$ and $G - S$ denote the graphs induced on S and $V \setminus S$, respectively. For a vertex subset $S \subseteq V$, we use $N_G(S)$ and $N_G[S]$ to denote the open and closed neighborhoods of S in G, respectively. That is, $N_G(S) = \{v \mid \{u, v\} \in E, u \in S\} \setminus S$ and $N_G[S] = N_G(S) \cup S$. In case $S = \{v\}$, we simply let $N(v) = N(S)$ and $N[v] = N[S]$. For a graph $G = (V, E)$ and an edge $e \in E$, we let G/e denote the graph obtained by contracting e in G. For $E' \subseteq \binom{V}{2}$, i.e. a subset of edges, we let $G + E'$ denote the (multi) graph obtained after adding the edges in E' to G, and we let G/E' denote the (multi) graph obtained after contracting the edges of E' in G. The girth of a graph is denoted by $\mathrm{girth}(G)$, its minimum degree by $\delta(G)$, and its maximum degree by $\Delta(G)$. A graph with no cycles has infinite girth.

A *path* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $\{v_i, v_{i+1}\}$ is an edge for all $0 \leq i < \ell$. A *cycle* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $\{v_i, v_{(i+1) \mod \ell+1}\}$ is an edge for all $0 \leq i \leq \ell$. Both a double edge and a self-loop are cycles. If P is a path from a vertex u to a vertex v in the graph G then we say that u and v are the end vertices of the path P and P is a (u, v)-path. For a path P, we use $V(P)$ and $E(P)$ to denote the sets of vertices and edges in the path P, respectively, and length of P is denoted by $|P|$ (i.e, $|P| = |V(P)|$). For a cycle C, we use $V(C)$ and $E(C)$ to denote the sets of vertices and edges in the cycle C, respectively, and the length of C, denoted by $|C|$, is $|V(C)|$. For a path or a cycle Q we use $N_G(Q)$ and $N_G[Q]$ to denote the sets $N_G(V(Q))$ and $N_G[V(Q)]$, respectively. For a collection of paths/cycles $\mathcal{Q}$, we use $|\mathcal{Q}|$ to denote the number of paths/cycles in $\mathcal{Q}$ and $V(\mathcal{Q})$ to denote the set $\bigcup_{Q \in \mathcal{Q}} V(Q)$. We say a path P in G is a *degree-two path* if all vertices in $V(P)$, including the end vertices of P, have degree exactly 2 in G. We say P is a *maximal degree-two path* if no proper superset of P also forms a degree-two path. We note that the notions of *walks* and *closed walks* are defined exactly as paths and cycles, respectively, except that their vertices need not be distinct. Finally, a *feedback vertex set* is a subset F of vertices such that $G - F$ is a forest.

Below we formally state some of the key results that will be used throughout the paper, starting with the classic Erdős-Pósa theorem [15].

▶ **Proposition 2** ([15]). *There exists a constant c' such that every (multi) graph either contains k vertex-disjoint cycles or it has a feedback vertex set of size at most $c'k\log k$.*

Observe that any (multi) graph $G=(V,E)$ whose feedback vertex set number is bounded by $c'k\log k$ has less than $(2c'k\log k+1)\cdot|V|$ edges (recall that we restrict the multiplicity of an edge to be 2). Indeed, letting F denote a feedback vertex set of minimum size, the worst case (in terms of $|E|$) is obtained when $G-F$ is a tree, which contains $|V|-|F|-1$ edges, and between every pair of vertices $v\in F$ and $u\in V$, there exists an edge of multiplicity 2. Thus, by Proposition 2, in case $|E|>(2c'k\log k+1)\cdot|V|$, the input instance is a yes-instance, and after we discard an arbitrary set of $|E|-(2c'k\log k+1)\cdot|V|$ edges, it remains a yes-instance. A simple operation which discards at least $|E|-(2c'k\log k+1)\cdot|V|$ edges and can be performed in time $\mathcal{O}(k\log k\cdot|V|)$.

▶ **Assumption 3.** *We assume that $|E|=\mathcal{O}(k\log k\cdot|V|)$.*

Now, we state our algorithmic version of Proposition 2. The proof partially builds upon the proof of the Erdős-Pósa theorem in the book [12], and it is given in the full version of the paper.

▶ **Theorem 4.** *There exists a constant c and a polynomial-space algorithm such that given a (multi) graph G and a non-negative integer k, in time $k^{\mathcal{O}(1)}\cdot|V|$ it either outputs k vertex-disjoint cycles or a feedback vertex set of size at most $ck\log k=r$.*

Next, we state two results relating to cycles of average and short lengths.

▶ **Proposition 5** ([1]). *Any (multi) graph $G=(V,E)$ on n vertices with average degree d contains a cycle of length at most $2\log_{d-1}n+2$.*

Itai and Rodeh [21] showed that given a (multi) graph $G=(V,E)$, an "almost" shortest cycle (if there is any) in G can be found in time $\mathcal{O}(|V|^2)$. To obtain a linear dependency on $|V|$ (given a small feedback vertex set), we prove the following result. A complete proof will appear in the full version of the paper.

▶ **Lemma 6.** *Given a (multi) graph $G=(V,E)$ and a feedback vertex set F of G, a shortest cycle (if there is any) in G can be found in time $\mathcal{O}(|F|\cdot(|V|+|E|))$.*

Finally, we state a result that will be used (in Lemma 11) to bound the size of a graph we obtain after performing simple preprocessing operations as well as repetitive removal of short cycles.

▶ **Proposition 7** ([34], Lemma 9). *Let $T=(V,E)$ be a forest on N vertices. Let $M'=\{\{i,j\}\in E\mid deg_T(i)=deg_T(j)=2\}$ and $L=\{a\in V\mid deg_T(a)\leq 1\}$. Then there exists $M\subseteq M'$ such that M is a matching and $|W|\geq\frac{N}{4}$ where $W=L\cup M$.*

3 Removing Leaves, Induced Paths, and Short Cycles

As is usually the case when dealing with cycles in a graph, we first define three rules which help getting rid of vertices of degree at most 2 as well as edges of multiplicity larger than 2. It is not hard to see that all three Reduction Rules A1, A2, and A3 are safe, i.e. they preserve solutions in the reduced graph.

▶ **Reduction Rule A1.** *Delete vertices of degree at most 1.*

▶ **Reduction Rule A2.** *If there is a vertex v of degree exactly 2 that is not incident to a self-loop, then delete v and connect its two (not necessarily distinct) neighbors by a new edge.*

▶ **Reduction Rule A3.** *If there is a pair of vertices u and v in V such that $\{u, v\}$ is an edge of multiplicity larger than 2, then reduce the multiplicity of the edge to 2.*

Observe that the entire process that applies these rules exhaustively can be done in time $\mathcal{O}(|V| + |E|) = \mathcal{O}(k \log k \cdot |V|)$. Indeed, in time $\mathcal{O}(|V|)$ we first remove the vertex-set of each maximal path between a leaf and a degree-two vertex. No subsequent application of Rule A2 or Rule A3 creates vertices of degree at most one. Now, we iterate over the set of degree-two vertices. For each degree-two vertex that is not incident to a self-loop, we apply Rule A2. Next, we iterate over E, and for each edge of multiplicity larger than two, we apply Rule A3. At this point, the only new degree-two vertices that can be created are vertices incident to exactly one edge, whose multiplicity is two. Therefore, during one additional phase where we exhaustively apply Rule A2, the only edges of multiplicity larger than two that can be created are self-loops. Thus, after one additional iteration over E, we can ensure that no rule among Rules A1, A2 and A3 is applicable.

Since these rules will be applied dynamically and iteratively, we define an operator, denoted by $\mathsf{reduce}(G)$, that takes as input a graph G and returns the (new) graph G' that results from an exhaustive application of Rules A1, A2 and A3.

▶ **Definition 8.** For a (multi) graph G, we let $G' = \mathsf{reduce}(G)$ denote the graph obtained after an exhaustive application of Reduction Rules A1, A2 and A3. $|\mathsf{reduce}(G)|$ denotes the number of vertices in $\mathsf{reduce}(G)$. Moreover, $\mathsf{img}(\mathsf{reduce}(G))$ denotes the pre-image of $\mathsf{reduce}(G)$, i.e. $\mathsf{img}(\mathsf{reduce}(G))$ is the set of vertices in G which are not deleted in $\mathsf{reduce}(G)$.

▶ **Observation 9.** *For a graph $G = (V, E)$ and a set $E' \subseteq \binom{V}{2}$ it holds that $|\mathsf{reduce}(G+E')| \leq |\mathsf{reduce}(G)| + 2|E'|$.*

The first step of our algorithm consists of finding, in time linear in $|V|$, a set S satisfying the conditions specified in Lemmata 10 and 11. Intuitively, S will contain vertices of "short" cycles in the input graph, where short will be defined later.

▶ **Lemma 10.** *Given a (multi) graph $G = (V, E)$ and two integers $k > 0$ and $g > 6$, there exists an $k^{\mathcal{O}(1)} \cdot |V|$-time algorithm that either finds k vertex-disjoint cycles in G or finds a (possibly empty) set $S \subseteq V$ such that girth$(\mathsf{reduce}(G - S)) > g$ and $|S| < gk$.*

Proof. We proceed by constructing such an algorithm. First, we apply the algorithm of Theorem 4 which outputs either k vertex-disjoint cycles or a feedback vertex set F of size at most $ck \log k = r$. In the former case we are done. In the latter case, i.e. the case where a feedback vertex set F is obtained, we apply the following procedure iteratively (initially, we set $S = \emptyset$):

(1) Apply Lemma 6 to find a shortest cycle C in $\mathsf{reduce}(G)$.
(2) If no cycle was found or $|C| > g$ then return S.
(3) Otherwise, i.e. if $|C| \leq g$, then add the vertices of C to S, delete those vertices from G to obtain G', set $G = G'$, and repeat from Step (1).

Note that if Step (3) is applied k times then we can terminate and return the corresponding k vertex-disjoint cycles in G. Hence, when the condition of Step (2) is satisfied, i.e. when the described procedure terminates, the size of S is at most $g(k-1) < gk$ and girth$(\mathsf{reduce}(G - S)) > g$. Since the algorithm of Theorem 4 runs in time $k^{\mathcal{O}(1)} \cdot |V|$, and each iteration of Steps (1)-(3) is performed in time $\mathcal{O}((k \log k)^2 \cdot |V|)$, we obtain the desired time complexity. ◀

▶ **Lemma 11.** *Given a (multi) graph $G = (V, E)$ and two integers $k > 0$ and $g > 6$, let S denote the set obtained after applying the algorithm of Lemma 10 (assuming no k vertex-disjoint cycles obtained). Then $|\mathsf{reduce}(G - S)| \leq (2ck \log k)^{1+\frac{6}{g-6}} + 3ck \log k$.*

Proof. Let $G' = (V', E') = \mathsf{reduce}(G - S)$ and $|V'| = n'$. First, recall that G admits a feedback vertex set of size at most $ck \log k = r$. Since Reduction Rules A1, A2 and A3 do not increase the feedback vertex set of the graph (see, e.g., [34], Lemma 1), G' also admits a feedback vertex set F of size at most r. Let T denote the induced forest on the remaining $N = n' - r$ vertices in G'. Moreover, from Lemma 10, we know that $\text{girth}(G') > g > 6$.

Next, we apply Proposition 7 to T to get W. Now with every element $a \in W$ we associate an unordered pair of vertices of F as follows. Assume $a \in L$, i.e. a is a vertex of degree 0 or 1. Since the degree of a is at least 3 in G', a has at least two neighbors in F. We pick two of these neighbors arbitrarily and associate them with a. We use $\{x_a, y_a\}$ to denote this pair. If $a = \{u, v\}$ is an edge from M then each of u and v has degree at least 3 in G' and each has at least one neighbor in F. We pick one neighbor for each and associate the pair $\{x_u, x_v\}$ with a. Note that since $\text{girth}(G') > 6$, $x_u \neq x_v$ and $x_a \neq y_a$.

We now construct a new multigraph $G^\star = (V^\star, E^\star)$ with vertex set $V^\star = F$ as follows. For every vertex $a \in W$ we include an edge in $E^\star$ between x_a and y_a, and for every edge $a = \{u, v\} \in W$ we include an edge in $E^\star$ between x_u and x_v. By Proposition 7, we know that W is of size at least $\frac{N}{4}$. It follows that $G^\star$ has at least $\frac{N}{4}$ edges and hence its average degree is at least $\frac{N}{2r}$ as $|V^\star| = ck \log k = r$. Note that if $G^\star$ has a cycle of length at most ℓ, then G' has a cycle of length at most 3ℓ, as any edge of the cycle in $G^\star$ can be replaced by a path of length at most 3 in G'. Combining this with the fact that $\text{girth}(G') > g > 6$, we conclude that $G^\star$ contains no self-loops or parallel edges. Hence $G^\star$ is a simple graph with average degree at least $\frac{N}{2r}$. By Proposition 5, $G^\star$ must have a cycle of length at most

$$2 \log_{\frac{N}{2r}-1} r + 2 = \frac{2 \log r}{\log(\frac{N}{2r} - 1)} + 2$$

which implies that G' must have a cycle of length at most

$$\frac{6 \log r}{\log(\frac{N}{2r} - 1)} + 6.$$

Finally, by using the fact that $\text{girth}(G') > g$ and substituting N and r, we get

$$\begin{aligned}
\frac{6 \log r}{\log(\frac{N}{2r} - 1)} + 6 > g \quad &\iff \log r > \frac{(g-6)}{6} \log\left(\frac{N - 2r}{2r}\right) \\
&\iff \log r > \frac{(g-6)}{6} \log(N - 2r) - \frac{(g-6)}{6} \log(2r) \\
&\iff \frac{\log r + \frac{(g-6)}{6} \log(2r)}{\frac{(g-6)}{6}} > \log(N - 2r) \\
&\implies \frac{\log(2r) + \frac{(g-6)}{6} \log(2r)}{\frac{(g-6)}{6}} > \log(N - 2r) \\
&\iff (1 + \frac{6}{g-6}) \log(2r) > \log(N - 2r) \\
&\iff (1 + \frac{6}{g-6}) \log(2ck \log k) > \log(n' - 3ck \log k) \\
&\iff (2ck \log k)^{1+\frac{6}{(g-6)}} + 3ck \log k > n'.
\end{aligned}$$

This completes the proof. ◀

The usefulness of Lemma 11 comes from the fact that by setting $g = \frac{48 \log k}{\log \log k} + 6$, we can guarantee that $|\mathsf{reduce}(G - S)| < 3ck \log k + 2ck \log^{1.5} k$, and therefore we can beat the $\mathcal{O}(k \log^2 k)$ bound. That is, we have the following consequence.

▶ **Corollary 12.** *Given a (multi) graph $G = (V, E)$ and an integer $k > 0$, let S denote the set obtained after applying the algorithm of Lemma 10 with $g = \frac{48 \log k}{\log \log k} + 6$ (assuming no k vertex-disjoint cycles obtained). Then $|\mathsf{reduce}(G - S)| \leq 3ck \log k + 2ck \log^{1.5} k$.*

Proof. By Lemma 11, $|\mathsf{reduce}(G - S)| \leq (2ck \log k)^{1+\frac{\log \log k}{8 \log k}} + 3ck \log k$. Assuming $k > \log k > c > 2$, we have $(2ck \log k)^{1+\frac{\log \log k}{8 \log k}} = (2ck \log k)(2ck \log k)^{\frac{\log \log k}{8 \log k}} \leq (2ck \log k)k^{\frac{4 \log \log k}{8 \log k}}$. Now note that $k^{\frac{4 \log \log k}{8 \log k}} \leq \log^{0.5} k$. Hence, $(2ck \log k)^{1+\frac{\log \log k}{8 \log k}} \leq 2ck \log k \log^{\frac{1}{2}} k \leq 2ck \log^{1.5} k$. This completes the proof. ◀

4 Bounding the Core of the Remaining Graph

At this point, we assume, without loss of generality, that we are given a graph $G = (V, E)$, a positive integer k, $g = \frac{48 \log k}{\log \log k} + 6$, and a set $S \subseteq V$ such that $\mathrm{girth}(\mathsf{reduce}(G - S)) > g$, $|S| < gk$, and $|\mathsf{reduce}(G - S)| \leq 3ck \log k + 2ck \log^{1.5} k$.

Even though the number of vertices in $\mathsf{reduce}(G - S)$ is bounded, the number of vertices in $G - S$ is unbounded. In what follows, we show how to bound the number of "objects" in $G - S$, where an object is either a vertex in $G - S$ or a degree-two path in $G - S$. The next lemma is a refinement extending a lemma by Lokshtanov et al. [31] (Lemma 5.2).

▶ **Lemma 13.** *Let $G = (V, E)$ be a (multi) graph and let $X \subseteq V$ be any subset of the vertices of G. Suppose there are more than $|X|^2(2|X| + 1)$ vertices in $G - X$ whose degree in $G - X$ is at most one. Then, there is either an isolated vertex w in $G - X$ or an edge $e \in E$ such that (G, k) is a yes-instance of* Cycle Packing *if and only if either $(G - \{w\}, k)$ or $(G/e, k)$ is a yes-instance. Moreover, there is an $\mathcal{O}(|X|^2 \cdot k \log k \cdot |V|)$-time algorithm that given G and X, outputs sets $V_X \subseteq V \setminus X$ and $E_X \subseteq E(G - X)$ such that, for the graph $G' = (G/E_X) - V_X$, it holds that (G, k) is a yes-instance of* Cycle Packing *if and only if (G', k) is a yes-instance of* Cycle Packing*, and $G' - X$ contains at most $|X|^2(2|X| + 1)$ vertices whose degree in $G' - X$ is at most one.*

Armed with Lemma 13, we are now ready to prove the following result. For a forest T, we let $T_{\leq 1}$, T_2, and $T_{\geq 3}$, denote the sets of vertices in T having degree at most one in T, degree exactly two in T, and degree larger than two in T, respectively. Moreover, we let $\mathcal{P}$ denote the set of all maximal degree-two paths in T.

▶ **Lemma 14.** *Let $G = (V, E)$, S, k, and g be as defined above. Let $R = \mathsf{img}(\mathsf{reduce}(G - S)) \subseteq (V \setminus S)$ denote the pre-image of $\mathsf{reduce}(G - S)$ in $G - S$. Then, $T = G - S - R$ is a forest and for every maximal degree-2 path in $\mathcal{P}$ there are at most two vertices on the path having neighbors in R (in the graph $G - S$). Moreover, in time $k^{\mathcal{O}(1)} \cdot |V|$, we can guarantee that $|T_{\leq 1}|$, $|\mathcal{P}|$, and $|T_{\geq 3}|$ are bounded by $k^{\mathcal{O}(1)}$.*

Proof. To see why $T = G - S - R$ must be a forest it is sufficient to note that for any cycle in $G - S$ at least one vertex from that cycle must be in $R = \mathsf{img}(\mathsf{reduce}(G - S))$ (see Figure 1). Recall that, since $\mathrm{girth}(\mathsf{reduce}(G - S)) > 6$, every vertex in R has degree at least 3 in $G - S$. Now assume there exists some path $P \in \mathcal{P}$ having exactly three (the same argument holds for any number) distinct vertices u, v and w (in that order) each having at least one neighbor in R (possibly the same neighbor). We show that the middle vertex v must have been in R,

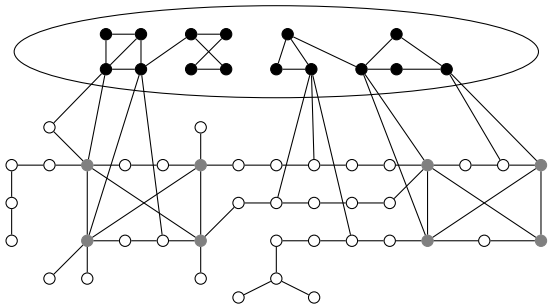


Figure 1 A graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white).

contradicting the fact that $T = G - S - R$. Consider the graph $G - S$ and apply Reduction Rules A1, A2 and A3 exhaustively (in $G - S$) on all vertices in the tree containing P except for u, v and w. Regardless of the order in which we apply the reduction rules, the path P will eventually reduce to a path on three vertices, namely u, v, and w. To see why v must be in R observe that even if the other two vertices have degree two in the resulting graph, after reducing them, v will have degree at least three (into R) and is therefore non-reducible.

Next, we bound the size of $T_{\leq 1}$, which implies a bound on the sizes of $T_{\geq 3}$ and $\mathcal{P}$. To do so, we simply invoke Lemma 13 by setting $X = S \cup R$. Since $|S| < gk$, $g = \frac{48 \log k}{\log \log k} + 6$ and $|R| \leq 3ck \log k + 2ck \log^{1.5} k$, we get that $|T_{\leq 1}| \leq |S \cup R|^2(2|S \cup R| + 1) = k^{\mathcal{O}(1)}$. Since in a forest, it holds that $|T_{\geq 3}| < |T_{\leq 1}|$, the bound on $|T_{\geq 3}|$ follows. Moreover, in a forest, it also holds that $|\mathcal{P}| < |T_{\leq 1}| + |T_{\geq 3}|$ – if we arbitrarily root each tree in the forest at a leaf, one end vertex of a path in $\mathcal{P}$ will be a parent of a different vertex from $T_{\leq 1} \cup T_{\geq 3}$ – the bound on $|\mathcal{P}|$ follows as well. ◀

5 Guessing Permutations

This section is devoted to proving the following lemma. Note that assuming the statement of the lemma, the only remaining task (to prove Theorem 1) is to develop an algorithm running in time $\mathcal{O}(2^{|V|} \cdot \text{poly}(|V|))$ and using polynomial space, which we present in Section 6.

▶ **Lemma 15.** *Given an instance (G, k) of* Cycle Packing*, we can in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$ and polynomial space compute $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ instances of* Cycle Packing *of the form (G', k), where the number of vertices in G' is bounded by $\mathcal{O}(k \log^{1.5} k)$, such that (G, k) is a yes-instance if and only if at least one of the instances (G', k) is a yes-instance.*[2]

Proof. We fix $g = \frac{48 \log k}{\log \log k} + 6$. Using Lemma 10, we first compute a set S in time $k^{\mathcal{O}(1)} \cdot |V|$. Then, we guess which vertices to delete from S – that is, which vertices do not participate in a solution – in time $\mathcal{O}(2^{gk}) = 2^{\mathcal{O}(\frac{k \log k}{\log \log k})}$. Here, guesses refer to different choices which lead to the construction of different instances of Cycle Packing that are returned at the end (recall that we are allowed to return up to $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ different instances). Combining Lemma 10 and Corollary 12, we now have a set $S \subseteq V$ such that $|S| = \mathcal{O}(\frac{k \log k}{\log \log k})$, and $|\mathsf{reduce}(G - S)| = \mathcal{O}(k \log^{1.5} k)$.

[2] In practice, to use polynomial space, we output the instances one-by-one.

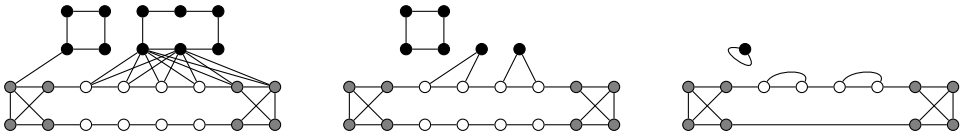


Figure 2 [Left] A graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white). [Center] the graph obtained after guessing vertices in S and their neighbors in a solution. [Right] Example of a reduced instance.

Applying Lemma 14 with $R = \mathsf{img}(\mathsf{reduce}(G-S)) \subseteq (V \setminus S)$, we get a forest $T = G-(S \cup R)$ such that for every maximal degree-two path in $\mathcal{P}$ there are at most two vertices on the path having neighbors in R (in the graph $G - S$). In addition, the size of R is bounded by $\mathcal{O}(k \log^{1.5} k)$, and the sizes $|T_{\leq 1}|$, $|\mathcal{P}|$ and $|T_{\geq 3}|$ are bounded by $k^{\mathcal{O}(1)}$ (see Figure 1).

For every vertex in S (which is assumed to participate in a solution), we now guess its two neighbors in a solution (see Figure 2). Note however that we only have a (polynomial in k) bound for $|S|$, $|R|$, $|T_{\leq 1}|$, $|\mathcal{P}|$ and $|T_{\geq 3}|$, but not for the length of paths in $\mathcal{P}$ and therefore not for the entire graph G. We let $Z_{\mathcal{P}}$ denote the set of vertices in $V(\mathcal{P})$ having neighbors in R. The size of $Z_{\mathcal{P}}$ is at most $2|\mathcal{P}|$. Moreover, we let $\mathcal{P}^\star$ denote the set of paths obtained after deleting $Z_{\mathcal{P}}$ from $\mathcal{P}$. Note that the size of $\mathcal{P}^\star$ is upper bounded by $|\mathcal{P}| + |Z_{\mathcal{P}}| \leq 3|\mathcal{P}|$, and that vertices in $V(\mathcal{P}^\star)$ are adjacent only to vertices in $V(\mathcal{P}^\star) \cup Z_{\mathcal{P}} \cup S$. Now, we create a set of "objects", $O = S \cup R \cup T_{\leq 1} \cup T_{\geq 3} \cup Z_{\mathcal{P}} \cup \mathcal{P}^\star$. We also denote $\widetilde{O} = O \setminus \mathcal{P}^\star$. We then guess, for each vertex in S, which two objects in O constitute its neighbors, denoted by $\ell(v)$ and $r(v)$, in a solution. It is possible that $\ell(v) = r(v)$. Since $|O| = k^{\mathcal{O}(1)}$, we can perform these guesses in $k^{\mathcal{O}(\frac{k \log k}{\log \log k})}$, or equivalently $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$, time. We can assume that if $\ell(v) \in \widetilde{O}$, then $\ell(v)$ is a neighbor of v, and otherwise v has a neighbor on the path $\ell(v)$, else the current guess is not correct, and we need not try finding a solution subject to it. The same claim holds for $r(v)$. If $\ell(v) = r(v) \in \widetilde{O}$, then $\{v, \ell(v)\}$ is an edge of multiplicity two, and otherwise if $\ell(v) = r(v)$, then v has (at least) two neighbors on the path $\ell(v)$.

Next, we fix some arbitrary order on $\mathcal{P}^\star$, and for each path in $\mathcal{P}^\star$, we fix some arbitrary orientation. We let $S^\star$ denote the multiset containing two occurrences of every vertex $v \in S$, denoted by v_ℓ and v_r. We guess an order of the vertices in $S^\star$. The time spent for guessing such an ordering is bounded by $|S|!$, which in turn is bounded by $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$. The ordering, assuming it is guessed correctly, satisfies the following conditions. For each path $P \in \mathcal{P}^\star$, we let $\ell(P)$ and $r(P)$ denote the sets of vertices $v \in S$ such that $\ell(v) \in V(P)$ and $r(v) \in V(P)$, respectively. Now, for any two vertices $u, v \in \ell(P)$, if $u_\ell < v_\ell$ according to the order that we guessed, then the neighbor $\ell(u)$ of u appears before the neighbor $\ell(v)$ of v on P. Similarly, for any two vertices $u, v \in r(P)$, if $u_r < v_r$, then $r(u)$ appears before $r(v)$ on P. Finally, for any two vertices $u \in \ell(P)$ and $v \in r(P)$, if $u_\ell < v_r$, then $\ell(u)$ appears before $r(v)$ on P, and otherwise $r(v)$ appears before $\ell(u)$ on P.

Given a correct guess of $\ell(v)$ and $r(v)$, for each v in S, as well as a correct guess of a permutation of $S^\star$, for each path in $\mathcal{P}^\star$, we let $\{x_v, y_v\}$ denote the two guessed neighbors of a vertex v in S. Note that if $\ell(v)$ ($r(v)$) is in $\widetilde{O}$ then $x_v = \ell(v)$ ($y_v = r(v)$). Otherwise, we assign neighbors to a vertex by a greedy procedure which agrees with the guessed permutation on $S^\star$; that is, for every path $P \in \mathcal{P}^\star$, we iterate over $\ell(P) \cup r(P)$ according to the guessed order, and for each vertex in it, assign its first neighbor on P that is after the last vertex that has already been assigned (if such a vertex does not exist, we determine that the current guess is incorrect and proceed to the next one). We let $X = \{x_v \mid v \in S\}$ and $Y = \{y_v \mid v \in S\}$. We also let E_S be the set of edges incident on a vertex in S, and we let $E' = \{\{x_v, y_v\} \mid v \in S\}$

denote the set of all pairs of guesses. Finally, to obtain an instance (G', k), we delete the vertex set $W = S \setminus (X \cup Y)$ from G, we delete the edge set E_S from G, we add instead the set of edges E', and finally we apply the reduce operator, i.e. $G' = \mathsf{reduce}((G - W - E_S) + E')$.

▶ **Claim 16.** *Let (G', k) be one of the instances generated by the above procedure. Then, the number of vertices in G' is bounded by $\mathcal{O}(k \log^{1.5} k)$.*

Proof. Recall that by Corollary 12, we know that $|\mathsf{reduce}(G - S)| = \mathcal{O}(k \log^{1.5} k)$. Moreover, we have $|E'| = |S| = \mathcal{O}(\frac{k \log k}{\log \log k})$. Combining Observation 9 with the fact that $G' = \mathsf{reduce}((G - W - E_S) + E')$, we get $|\mathsf{reduce}((G - W - E_S) + E')| \leq |\mathsf{reduce}(G - W - E_S)| + 2|E'|$. Since in $G - W - E_S$ all vertices of S have degree zero, $|\mathsf{reduce}(G - W - E_S)| \leq |\mathsf{reduce}(G - S)|$. Hence, we conclude that $|\mathsf{reduce}((G - W - E_S) + E')| = \mathcal{O}(k \log^{1.5} k)$, as needed. ◀

▶ **Claim 17.** *(G, k) is a yes-instance if and only if at least one of the generated instances (G', k) is a yes-instance.*

Proof. Assume that (G, k) is a yes-instance and let $\mathcal{C} = \{C_1, C_2, \ldots\}$ be an optimal cycle packing, i.e set of maximum size of vertex-disjoint cycles, in G. Note that if no cycle in $\mathcal{C}$ intersects with S then $\mathcal{C}$ is also an optimal cycle packing in $G - S$. By the safeness of our reduction rules, $\mathcal{C}$ is also an optimal cycle packing in $\mathsf{reduce}(G - S)$. Since we generate one instance for every possible intersection between an optimal solution and S, the case where no vertex from S is picked corresponds to the instance (G', k), with $G' = \mathsf{reduce}(G - S)$. Hence, in what follows we assume that some cycles in $\mathcal{C}$ intersect with S. Consider any cycle C which intersects with S and let $P_C = \{u_0, u_1, \ldots, u_f\}$ denote any path on this cycle such that $u_0, u_f \notin S$ but $u_i \in S$ for $0 < i < f$. We claim that, for some G', all such paths will be replaced by edges of the form $\{u_0, u_f\}$ in $\mathsf{reduce}((G - W - E_S) + E')$. Again, due to our exhaustive guessing, for some G' we would have guessed, for each i, $\ell(u_i) = u_{i-1}$ and $r(u_i) = u_{i+1}$. Consequently, $P_C \setminus \{u_0, u_f\}$ is a degree-two path in $(G - W - E_S) + E'$ and therefore an edge in $\mathsf{reduce}((G - W - E_S) + E')$. Using similar arguments, it is easy to show that if C is completely contained in S then this cycle is contained in G' as a loop on some vertex of the cycle.

For the other direction, let (G', k) be a yes-instance and let $\mathcal{C}' = \{C'_1, C'_2, \ldots\}$ be an optimal cycle packing in G'. We assume, without loss of generality, that $\mathcal{C}'$ is a cycle packing in $(G - W - E_S) + E'$, as one can trace back all reduction rules to obtain the graph $(G - W - E_S) + E'$. If no cycle in $\mathcal{C}'$ uses an edge $\{u_0, u_f\} \in E'$ then we are done, as $(G - W - E_S)$ is a subgraph of G. Otherwise, we claim that all such edges either exist in G or can be replaced by vertex disjoint paths $P = \{u_0, u_1, \ldots, u_f\}$ (on at least three vertices) in G such that $u_i \in S$ for $0 < i < f$. If either u_0 or u_f is in $X \cup Y \subseteq S$ then the former case holds. It remains to prove the latter case. Recall that for every vertex in S we guess its two neighbors from $O = S \cup R \cup T_{\leq 1} \cup T_{\geq 3} \cup Z_{\mathcal{P}} \cup \mathcal{P}^\star$. Hence, if $\{u_0, u_f\} \subseteq \widetilde{O} = O \setminus \mathcal{P}^\star$ then one can easily find a path (or singleton) in $G[S]$ to replace this edge by simply backtracking the neighborhood guesses. Now assume that $\{u_0, u_f\} \not\subseteq \widetilde{O}$ and recall that no vertex in a path in $\mathcal{P}^\star$ can have neighbors in R. Hence, any cycle containing such an edge must intersect with S (in G). Assuming we have correctly guessed the neighbors of vertices in S (as well as a permutation for $\mathcal{P}^\star$), we can again replace this edge with a path in S. ◀

Combining Claims 16 and 17 concludes the proof of the theorem. ◀

6 Dynamic Programming and Inclusion-Exclusion

Finally, we give an exact exponential-time algorithm for Cycle Packing. For this purpose, we use DP and the principle of inclusion-exclusion, inspired by the work of Nederlof [32]. Due to space constraints, the details are given in the full version of the paper.

▶ **Lemma 18.** *There exists a (deterministic) polynomial-space algorithm that in time* $\mathcal{O}(2^{|V|} \cdot \text{poly}(|V|))$ *solves* Cycle Packing. *In case a solution exists, it also outputs a solution.*

We would like to mention that if one does not care about polynomial space, then Lemma 18 can be obtained by a straightforward dynamic programming on subsets.

7 Conclusion

In this paper we have beaten the best known $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time algorithm for Cycle Packing that is a consequence of the Erdős-Pósa theorem. For this purpose, we developed a deterministic algorithm that solves Cycle Packing in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$. Two additional advantageous properties of our algorithm is that its space complexity is polynomial in the input size and that in case a solution exists, it outputs a solution (in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$). Our technique relies on combinatorial arguments that may be of independent interest. These arguments allow us to translate any input instance of Cycle Packing into $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ instances of Cycle Packing whose sizes are small and can therefore be solved efficiently.

It remains an intriguing open question to discover the "true" running time, under reasonable complexity-theoretic assumptions, in which one can solve Cycle Packing on general graphs. In particular, we would like to pose the following question: Does there exist a $2^{\mathcal{O}(k \log k)} \cdot |V|^{\mathcal{O}(1)}$-time algorithm for Cycle Packing? This is true for graphs of bounded maximum degree as one can easily bound the number of vertices by $\mathcal{O}(k \log k)$ and then apply Lemma 18. Moreover, Bodlaender et al. [4] proved that this is also true in case one seeks k edge-disjoint cycles rather than k vertex-disjoint cycles. On the negative side, recall that (for general graphs) the bound $f(k) = \mathcal{O}(k \log k)$ in the Erdős-Pósa theorem is essentially tight, and that it is unlikely that Cycle Packing is solvable in time $2^{o(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ [11]. However, we do not rule out the existence of an algorithm solving Cycle Packing in time $2^{\mathcal{O}(fvs)} \cdot |V|^{\mathcal{O}(1)}$. Thus, the two most natural attempts to obtain a $2^{\mathcal{O}(k \log k)} \cdot |V|^{\mathcal{O}(1)}$-time algorithm – either replacing the bound $\mathcal{O}(k \log k)$ in the Erdős-Pósa theorem by $\mathcal{O}(k)$ or speeding-up the computation based on DP to run in time $2^{\mathcal{O}(tw)} \cdot |V|^{\mathcal{O}(1)}$ – lead to a dead end.

Acknowledgements. We would like to thank the reviewers for several suggestions and insightful remarks that have improved the presentation of the paper.

References

1 Noga Alon, Shlomo Hoory, and Nathan Linial. The Moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002. doi:10.1007/s003730200002.

2 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernel bounds for path and cycle problems. *Theor. Comput. Sci.*, 511:117–136, 2013. doi:10.1016/j.tcs.2012.09.006.

3 Hans L. Bodlaender, Eelko Penninkx, and Richard B. Tan. A linear kernel for the k-disjoint cycle problem on planar graphs. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation, 19th International Symposium, ISAAC 2008,*

Gold Coast, Australia, December 15-17, 2008. Proceedings, volume 5369 of *Lecture Notes in Computer Science*, pages 306–317. Springer, 2008. doi:10.1007/978-3-540-92182-0_29.

4 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.

5 H. L. Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994.

6 Chandra Chekuri and Julia Chuzhoy. Large-treewidth graph decompositions and applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 291–300. ACM, 2013. doi:10.1145/2488608.2488645.

7 Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *J. ACM*, 63(5):40:1–40:65, 2016.

8 Julia Chuzhoy. Excluded grid theorem: Improved and simplified. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 645–654. ACM, 2015. doi:10.1145/2746539.2746551.

9 Julia Chuzhoy. Excluded grid theorem: Improved and simplified (invited talk). In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPIcs*, pages 31:1–31:1. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.SWAT.2016.31.

10 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*. Springer, 2015.

11 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

12 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

13 Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Catalan structures and dynamic programming in h-minor-free graphs. *J. Comput. Syst. Sci.*, 78(5):1606–1622, 2012. doi:10.1016/j.jcss.2012.02.004.

14 R. Downey and M. Fellows. *Fundamentals of parameterized complexity*. Springer, 2013.

15 P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canad. J. Math.*, 17:347–352, 1965.

16 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 748–759. SIAM, 2011. doi:10.1137/1.9781611973082.59.

17 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 503–510. SIAM, 2010. doi:10.1137/1.9781611973075.43.

18 Zachary Friggstad and Mohammad R. Salavatipour. Approximability of packing disjoint cycles. *Algorithmica*, 60(2):395–400, 2011. doi:10.1007/s00453-009-9349-5.

19 Martin Grohe and Magdalena Grüber. Parameterized approximability of the disjoint cycle problem. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2007. doi:10.1007/978-3-540-73420-8_33.

20 R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

21 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. doi:10.1137/0207033.

22 Felix Joos. Parity linkage and the erdős-pósa property of odd cycles through prescribed vertices in highly connected graphs. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science – 41st International Workshop, WG 2015, Garching, Germany, June 17-19, 2015, Revised Papers*, volume 9224 of *Lecture Notes in Computer Science*, pages 339–350. Springer, 2015. doi:10.1007/978-3-662-53174-7_24.

23 Naonori Kakimura and Ken-ichi Kawarabayashi. Half-integral packing of odd cycles through prescribed vertices. *Combinatorica*, 33(5):549–572, 2013. doi:10.1007/s00493-013-2865-6.

24 Naonori Kakimura, Ken-ichi Kawarabayashi, and Dániel Marx. Packing cycles through prescribed vertices. *J. Comb. Theory, Ser. B*, 101(5):378–381, 2011. doi:10.1016/j.jctb.2011.03.004.

25 Ken-ichi Kawarabayashi and Yusuke Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. *J. Comb. Theory, Ser. B*, 102(4):1020–1034, 2012. doi:10.1016/j.jctb.2011.12.001.

26 Ken-ichi Kawarabayashi, Daniel Král', Marek Krcál, and Stephan Kreutzer. Packing directed cycles through a specified vertex set. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 365–377. SIAM, 2013. doi:10.1137/1.9781611973105.27.

27 Ken-ichi Kawarabayashi and Atsuhiro Nakamoto. The erdos-pósa property for vertex- and edge-disjoint odd cycles in graphs on orientable surfaces. *Discrete Mathematics*, 307(6):764–768, 2007. doi:10.1016/j.disc.2006.07.008.

28 Ken-ichi Kawarabayashi and Bruce A. Reed. Highly parity linked graphs. *Combinatorica*, 29(2):215–225, 2009. doi:10.1007/s00493-009-2178-y.

29 Ken-ichi Kawarabayashi and Paul Wollan. Non-zero disjoint cycles in highly connected group labelled graphs. *J. Comb. Theory, Ser. B*, 96(2):296–301, 2006. doi:10.1016/j.jctb.2005.08.001.

30 Michael Krivelevich, Zeev Nutov, Mohammad R. Salavatipour, Jacques Yuster, and Raphael Yuster. Approximation algorithms and hardness results for cycle packing problems. *ACM Trans. Algorithms*, 3(4):48, 2007. doi:10.1145/1290672.1290685.

31 D. Lokshtanov, F. Panolan, M. S. Ramanujan, and S. Saurabh. Lossy kernelization. *arXiv:1604.04111v2*, to appear in STOC 2017.

32 Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.

33 M. Pontecorvi and Paul Wollan. Disjoint cycles intersecting a set of vertices. *J. Comb. Theory, Ser. B*, 102(5):1134–1141, 2012. doi:10.1016/j.jctb.2012.05.004.

34 Venkatesh Raman, Saket Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Trans. Algorithms*, 2(3):403–415, 2006. doi:10.1145/1159892.1159898.

35 Dieter Rautenbach and Bruce A. Reed. The Erdős-Pósa property for odd cycles in highly connected graphs. *Combinatorica*, 21(2):267–278, 2001. doi:10.1007/s004930100024.

36 Bruce A. Reed. Mangoes and blueberries. *Combinatorica*, 19(2):267–296, 1999. doi:10.1007/s004930050056.

37 Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.

38 Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

39 Aleksandrs Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM J. Discrete Math.*, 24(1):146–157, 2010. doi:10.1137/070697781.

40 Carsten Thomassen. The Erdős-Pósa property for odd cycles in graphs of large connectivity. *Combinatorica*, 21(2):321–333, 2001. doi:10.1007/s004930100028.

An Efficient Strongly Connected Components Algorithm in the Fault Tolerant Model*†

Surender Baswana[1], Keerti Choudhary[2], and Liam Roditty[3]

1 **Department of Computer Science and Engineering, IIT Kanpur, Kanpur, India**
sbaswana@cse.iitk.ac.in
2 **Department of Computer Science and Engineering, IIT Kanpur, Kanpur, India**
keerti@cse.iitk.ac.in
3 **Department of Computer Science, Bar Ilan University, Ramat Gan, Israel**
liam.roditty@biu.ac.il

Abstract

In this paper we study the problem of maintaining the strongly connected components of a graph in the presence of failures. In particular, we show that given a directed graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, and an integer value $k \geq 1$, there is an algorithm that computes in $O(2^k n \log^2 n)$ time for any set F of size at most k the strongly connected components of the graph $G \setminus F$. The running time of our algorithm is almost optimal since the time for outputting the SCCs of $G \setminus F$ is at least $\Omega(n)$. The algorithm uses a data structure that is computed in a preprocessing phase in polynomial time and is of size $O(2^k n^2)$.

Our result is obtained using a new observation on the relation between strongly connected components (SCCs) and reachability. More specifically, one of the main building blocks in our result is a restricted variant of the problem in which we only compute strongly connected components that intersect a certain path. Restricting our attention to a path allows us to implicitly compute reachability between the path vertices and the rest of the graph in time that depends logarithmically rather than linearly in the size of the path. This new observation alone, however, is not enough, since we need to find an efficient way to represent the strongly connected components using paths. For this purpose we use a mixture of old and classical techniques such as the heavy path decomposition of Sleator and Tarjan [29] and the classical Depth-First-Search algorithm. Although, these are by now standard techniques, we are not aware of any usage of them in the context of dynamic maintenance of SCCs. Therefore, we expect that our new insights and mixture of new and old techniques will be of independent interest.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases Fault tolerant, Directed graph, Strongly connected components

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.72

1 Introduction

Computing the strongly connected components (SCCs) of a directed graph $G = (V, E)$, where $n = |V|$ and $m = |E|$, is one of the most fundamental problems in computer science. There are several classical algorithms for computing the SCCs in $O(m + n)$ time that are taught in any standard undergraduate algorithms course [9].

* Full version of this article is available at https://arxiv.org/abs/1610.04010.
† This research was partially supported by Israel Science Foundation (ISF) and University Grants Commission (UGC) of India. The research of the second author was partially supported by Google India under the Google India PhD Fellowship Award.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 72; pp. 72:1–72:15


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

In this paper we study the following natural variant of the problem in dynamic graphs. What is the fastest algorithm to compute the SCCs of $G \setminus F$, where F is any set of edges or vertices. The algorithm can use a polynomial size data structure computed in polynomial time for G during a preprocessing phase.

The main result of this paper is:

▶ **Theorem 1.** *There is an algorithm that computes the SCCs of $G \setminus F$, for any set F of k edges or vertices, in $O(2^k n \log^2 n)$ time. The algorithm uses a data structure of size $O(2^k n^2)$ computed in $O(2^k n^2 m)$ time for G during a preprocessing phase.*

Since the time for outputting the SCCs of $G \setminus F$ is at least $\Omega(n)$, the running time of our algorithm is optimal (up to a polylogarithmic factor) for any fixed value of k.

This dynamic model is usually called the fault tolerant model and its most important parameter is the time that it takes to compute the output in the presence of faults. It is an important theoretical model as it can be viewed as a restriction of the deletion only (decremental) model in which edges (or vertices) are deleted one after another and queries are answered between deletions. The fault tolerant model is especially useful in cases where the worst case update time in the more general decremental model is high.

There is wide literature on the problem of decremental SCCs. Recently, in a major breakthrough, Henzinger, Krinninger and Nanongkai [18] presented a randomized algorithm with $O(mn^{0.9+o(1)})$ total update time and broke the barrier of $\Omega(mn)$ for the problem. Even more recently, Chechik et al. [7] obtained an improved total running time of $O(m\sqrt{n \log n})$.

However, these algorithms and in fact all the previous algorithms have an $\Omega(m)$ worst case update time for a single edge deletion. This is not a coincidence. Recent developments in conditional lower bounds by Abboud and V. Williams [1] and by Henzinger, Krinninger, Nanongkai and Saranurak [19] showed that unless a major breakthrough happens, the worst case update time of a single operation in any algorithm for decremental SCCs is $\Omega(m)$. Therefore, in order to obtain further theoretical understanding on the problem of decremental SCCs, and in particular on the worst case update time it is only natural to focus on the restricted dynamic model of fault tolerant.

In the recent decade several different researchers used the fault tolerant model to study the worst case update time per operation for dynamic connectivity in undirected graphs. Pătraşcu and Thorup [26] presented connectivity algorithms that support edge deletions in this model. Their result was improved by the recent polylogarithmic worst case update time algorithm of Kapron, King and Mountjoy [21]. Duan and Pettie [13, 14] used this model to obtain connectivity algorithms that support vertex deletions.

In directed graphs, very recently, Georgiadis, Italiano and Parotsidis [16] considered the problem of SCCs but only for a single edge or a single vertex failure, that is $|F| = 1$. They showed that it is possible to compute the SCCs of $G \setminus \{e\}$ for any $e \in E$ (or of $G \setminus \{v\}$ for any $v \in V$) in $O(n)$ time using a data structure of size $O(n)$ that was computed for G in a preprocessing phase in $O(m + n)$ time. Our result is the first generalized result for any constant size F. This comes with the price of an extra $O(\log^2 n)$ factor in the running time, a slower preprocessing time and a larger data structure. In [16], Georgiadis, Italiano and Parotsidis also considered the problem of answering strong connectivity queries after one failure. They show construction of an $O(n)$ size oracle that can answer in constant time whether any two given vertices of the graph are strongly connected after failure of a single edge or a single vertex.

In a previous work [2] we considered the problem of finding a sparse subgraph that preserves single source reachability. More specifically, given a directed graph $G = (V, E)$ and

a vertex $s \in V$, a subgraph H of G is said to be a k-Fault Tolerant Reachability Subgraph (k-FTRS) for G if for any set F of at most k edges (or vertices), a vertex $v \in V$ is reachable from s in $G \setminus F$ if and only if v is reachable from s in $H \setminus F$. In [2] we proved that there exists a k-FTRS for s with at most $2^k n$ edges.

Using the k-FTRS structure, it is relatively straightforward to obtain a data structure that, for any pair of vertices $u, v \in V$ and any set F of size k, answers in $O(2^k n)$ time queries of the form:

"*Are u and v in the same SCC of $G \setminus F$?*"

The data structure consists of a k-FTRS for every $v \in V$. It is easy to see that u and v are in the same SCC of $G \setminus F$ if and only if v is reachable from u in k-FTRS$(u) \setminus F$ and u is reachable from v in k-FTRS$(v) \setminus F$. So the query can be answered by checking, using graph traversals, whether v is reachable from u in k-FTRS$(u) \setminus F$ and whether u is reachable from v in k-FTRS$(v) \setminus F$. The cost of these two graph traversals is $O(2^k n)$. The size of the data structure is $O(2^k n^2)$.

This problem, however, is much easier since the vertices in the query reveal which two k-FTRS we need to scan. In the challenge that we address in this paper *all* the SCCs of $G \setminus F$, for an arbitrary set F, have to be computed. However, using the same data structure as before, it is not really clear a-priori which of the k-FTRS we need to scan.

We note that our algorithm uses the k-FTRS which seems to be an essential tool but is far from being a sufficient one and more involved ideas are required. As an example to such a relation between a new result and an old tool one can take the deterministic algorithm of Łącki [23] for decremental SCCs in which the classical algorithm of Italiano [20] for decremental reachability trees in directed acyclic graphs is used. The main contribution of Łącki [23] is a new graph decomposition that made it possible to use Italiano's algorithm [20] efficiently.

1.1 An overview of our result

We obtain our $O(2^k n \log^2 n)$-time algorithm using several new ideas. Interestingly, one of the main building blocks is the following restricted variant of the problem.

> *Given any set F of k failed edges and any path P which is intact in $G \setminus F$, output all the SCCs of $G \setminus F$ that intersect with P (i.e. contain at least one vertex of P).*

To solve this restricted version, we implicitly solve the problem of reachability from x (and to x) in $G \setminus F$, for each $x \in P$. Though it is trivial to do so in time $O(2^k n|P|)$ using k-FTRS of each vertex on P, our goal is to preform this computation in $O(2^k n \log n)$ time, that is, in running time that is *independent* of the length of P (up to a logarithmic factor). For this we use a careful insight into the structure of reachability between P and V. Specifically, if $v \in V$ is reachable from $x \in P$, then v is also reachable from any predecessor of x on P, and if v is not reachable from x, then it cannot be reachable from any successor of x as well. Let w be any vertex on P, and let A be the set of vertices reachable from w in $G \setminus F$. Then we can split P at w to obtain two paths: P_1 and P_2. We already know that all vertices in P_1 have a path to A, so for P_1 we only need to focus on set $V \setminus A$. Also the set of vertices reachable from any vertex on P_2 must be a subset of A, so for P_2 we only need to focus on set A. This suggests a divide-and-conquer approach which along with some more insight into the structure of k-FTRS helps us to design an efficient algorithm for computing all the SCCs that intersect P.

In order to use the above result to compute all the SCCs of $G \setminus F$, we need a clever partitioning of G into a set of vertex disjoint paths. A Depth-First-Search (DFS) tree plays a crucial role here as follows. Let P be any path from root to a leaf node in a DFS tree T. If we compute the SCCs intersecting P and remove them, then the remaining SCCs must be contained in subtrees hanging from path P. So to compute the remaining SCCs we do not need to work on the entire graph. Instead, we need to work on each subtree. In order to pursue this approach efficiently, we need to select path P in such a manner that the subtrees hanging from P are of small size. The heavy path decomposition of Sleator and Tarjan [29] helps to achieve this objective. [1]

Our algorithm and data structure can be extended to support insertions as well. More specifically, we can report the SCCs of a graph that is updated by insertions and deletions of k edges in the same running time.

1.2 Related work

The problem of maintaining the SCCs of a graph was studied in the decremental model. In this model the goal is to maintain the SCCs of a graph whose edges are being deleted by an adversary. The main parameters in this model are the worst case update time per an edge deletion and the total update from the first edge deletion until the last. Frigioni et al.[15] presented an algorithm that has an *expected* total update time of $O(mn)$ if all the deleted edges are chosen at random. Roditty and Zwick [28] presented a Las-Vegas algorithm with an *expected* total update time of $O(mn)$ and *expected* worst case update time for any single edge deletion of $O(m)$. Łącki [23] presented a deterministic algorithm with a total update time of $O(mn)$, and thus solved the open problem posed by Roditty and Zwick in [28]. However, the worst case update time per a single edge deletion of his algorithm is $O(mn)$. Roditty [27] improved the worst case update time of a single edge deletion to $O(m \log n)$. Recently, in a major breakthrough, Henzinger, Krinninger and Nanongkai [18] presented a randomized algorithm with $O(mn^{0.9+o(1)})$ total update time. Very recently, Chechik et al. [7] obtained a total update time of $O(m\sqrt{n \log n})$. Note that all the previous works on decremental SCC are with $\Omega(m)$ worst case update time. Whereas, our result directly implies $O(n \log^2 n)$ worst case update time as long as the total deletion length is constant.

Most of the previous work in the fault tolerant model is on variants of the shortest path problem. Demetrescu, Thorup, Chowdhury and Ramachandran [10] designed an $O(n^2 \log n)$ size data structure that can report the distance from u to v avoiding x for any $u, v, x \in V$ in $O(1)$ time. Bernstein and Karger [3] improved the preprocessing time of [10] to $O(mn \text{ polylog } n)$. Duan and Pettie [12] designed such a data structure for two vertex faults of size $O(n^2 \log n)$. Weimann and Yuster [31] considered the question of optimizing the preprocessing time using Fast Matrix Multiplication (FMM) for graphs with integer weights from the range $[-M, M]$. Grandoni and Vassilevska Williams [17] improved the result of [31] based on a novel algorithm for computing all the replacement paths from a given source vertex in the same running time as solving APSP in directed graphs.

For the problem of single source shortest paths Parter and Peleg [25] showed that for unweighted graphs there is a subgraph with $O(n^{3/2})$ edges that supports one fault. They also showed a matching lower bound. Recently, Parter [24] extended this result to two faults with $O(n^{5/3})$ edges for undirected graphs. She also showed a lower bound of $\Omega(n^{5/3})$.

[1] We note that the heavy path decomposition was also used in the fault tolerant model in STACS'10 paper of [22], but in a completely different way and for a different problem.

Baswana and Khanna [22] showed that there is a subgraph with $O(n \log n)$ edges that preserves the distances from s up to a multiplicative stretch of 3 upon failure of any single vertex. For the case of edge failures, sparse fault tolerant subgraphs exist for general k. Bilò et al. [4] showed that we can compute a subgraph with $O(kn)$ edges that preserves distances from s up to a multiplicative stretch of $(2k+1)$ upon failure of any k edges. They also showed that we can compute a data structure of $O(kn \log^2 n)$ size that is able to report the $(2k+1)$-stretched distance from s in $O(k^2 \log^2 n)$ time.

The questions of finding graph spanners, approximate distance oracles and compact routing schemes in the fault tolerant model were studied in [11, 8, 5, 6].

1.3 Organization of the paper

We describe notations, terminologies, some basic properties of DFS, heavy-path decomposition, and k-FTRS in Section 2. In Section 3, we describe the fault tolerant algorithm for computing the strongly connected components intersecting any path. We present our main algorithm for handling k failures in Section 4. The details on how to extend our algorithm and data structure to support insertions as well is provided in the full version.

2 Preliminaries

Let $G = (V, E)$ denote the input directed graph on $n = |V|$ vertices and $m = |E|$ edges. We assume that G is strongly connected, since if it is not the case, then we may apply our result to each strongly connected component of G. We first introduce some notations that will be used throughout the paper.

- T: A DFS tree of G.
- $T(v)$: The subtree of T rooted at a vertex v.
- $Path(a, b)$: The tree path from a to b in T. Here a is assumed to be an ancestor of b.
- $depth(Path(a, b))$: The depth of vertex a in T.
- G^R: The graph obtained by reversing all the edges in graph G.
- $H(A)$: The subgraph of a graph H induced by the vertices of subset A.
- $H \setminus F$: The graph obtained by deleting the edges in set F from graph H.
- In-Edges(v, H): The set of all incoming edges to v in graph H.
- $P[a, b]$: The subpath of path P from vertex a to vertex b, assuming a and b are in P and a precedes b.
- $P::Q$: The path formed by concatenating paths P and Q in G. Here it is assumed that the last vertex of P is the same as the first vertex of Q.

Our algorithm for computing SCCs in a fault tolerant environment crucially uses the concept of a k-fault tolerant reachability subgraph (k-FTRS) which is a sparse subgraph that preserves reachability from a given source vertex even after the failure of at most k edges in G. A k-FTRS is formally defined as follows.

▶ **Definition 2** (k-FTRS)**.** Let $s \in V$ be any designated source. A subgraph H of G is said to be a k-Fault Tolerant Reachability Subgraph (k-FTRS) of G with respect to s if for any subset $F \subseteq E$ of k edges, a vertex $v \in V$ is reachable from s in $G \setminus F$ if and only if v is reachable from s in $H \setminus F$.

In [2], we present the following result for the construction of a k-FTRS for any $k \geq 1$.

▶ **Theorem 3** ([2]). *There exists an $O(2^k mn)$ time algorithm that for any given integer $k \geq 1$, and any given directed graph G on n vertices, m edges and a designated source vertex s, computes a k-FTRS for G with at most $2^k n$ edges. Moreover, the in-degree of each vertex in this k-FTRS is bounded by 2^k.*

Our algorithm will require the knowledge of the vertices reachable from a vertex v as well as the vertices that can reach v. So we define a k-FTRS of both the graphs – G and G^R with respect to any source vertex v as follows.

- $\mathcal{G}(v)$: The k-FTRS of graph G with v as source obtained by Theorem 3.
- $\mathcal{G}^R(v)$: The k-FTRS of graph G^R with v as source obtained by Theorem 3.

The following lemma states that the subgraph of a k-FTRS induced by $A \subset V$ can serve as a k-FTRS for the subgraph $G(A)$ given that A satisfies certain properties.

▶ **Lemma 4.** *Let s be any designated source and H be a k-FTRS of G with respect to s. Let A be a subset of V containing s such that every path from s to any vertex in A is contained in $G(A)$. Then $H(A)$ is a k-FTRS of $G(A)$ with respect to s.*

Proof. Let F be any set of at most k failing edges, and v be any vertex reachable from s in $G(A) \setminus F$. Since v is reachable from s in $G \setminus F$ and H is a k-FTRS of G, so v must be reachable from s in $H \setminus F$ as well. Let P be any path from s to v in $H \setminus F$. Then (i) all edges of P are present in H and (ii) none of the edges of F appear on P. Since it is already given that every path from s to any vertex in A is contained in $G(A)$, therefore, P must be present in $G(A)$. So every vertex of P belongs to A. This fact combined with the inferences (i) and (ii) implies that P must be present in $H(A) \setminus F$. Hence $H(A)$ is k-FTRS of $G(A)$ with respect to s. ◀

The next lemma is an adaptation of Lemma 10 from Tarjan's classical paper on Depth First Search [30] to our needs (for proof see the full version).

▶ **Lemma 5.** *Let T be a DFS tree of G. Let $a, b \in V$ be two vertices without any ancestor-descendant relationship in T, and assume that a is visited before b in the DFS traversal of G corresponding to tree T. Every path from a to b in G must pass through a common ancestor of a and b in T.*

2.1 A heavy path decomposition

The heavy path decomposition of a tree was designed by Sleator and Tarjan [29] in the context of dynamic trees. This decomposition has been used in a variety of applications since then. Given any rooted tree T, this decomposition splits T into a set $\mathcal{P}$ of vertex disjoint paths with the property that any path from the root to a leaf node in T can be expressed as a concatenation of at most $\log n$ subpaths of paths in $\mathcal{P}$. This decomposition is carried out as follows. Starting from the root, we follow the path downward such that once we are at a node, say v, the next node traversed is the child of v in T whose subtree is of maximum size, where the size of a subtree is the number of nodes it contains. We terminate upon reaching a leaf node. Let P be the path obtained in this manner. If we remove P from T, we are left with a collection of subtrees each of size at most $n/2$. Each of these trees hangs from P through an edge in T. We carry out the decomposition of these trees recursively. The following lemma is immediate from the construction of a heavy path decomposition.

▶ **Lemma 6.** *For any vertex $v \in V$, the number of paths in $\mathcal{P}$ which start from either v or an ancestor of v in T is at most $\log n$.*

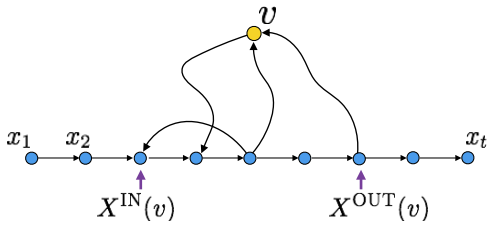


Figure 1 Depiction of $X^{\text{IN}}(v)$ and $X^{\text{OUT}}(v)$ for a vertex v whose SCC intersects X.

We now introduce the notion of ancestor path.

▶ **Definition 7.** A path $Path(a_1, b_1) \in \mathcal{P}$ is said to be an ancestor path of $Path(a_2, b_2) \in \mathcal{P}$, if a_1 is an ancestor of a_2 in T.

In this paper, we describe the algorithm for computing SCCs of graph G after any k edge failures. Vertex failures can be handled by simply splitting each vertex v into edge (v_{in}, v_{out}), where the incoming and outgoing edges of v are directed to v_{in} and from v_{out}, respectively.

3 Computation of SCCs intersecting a given path

Let F be a set of at most k failing edges, and $X = (x_1, x_2, \ldots, x_t)$ be any path in G from x_1 to x_t which is intact in $G \setminus F$. In this section, we present an algorithm that outputs in $O(2^k n \log n)$ time the SCCs of $G \setminus F$ that intersect X.

For each $v \in V$, let $X^{\text{IN}}(v)$ be the vertex of X of minimum index (if exists) that is reachable from v in $G \setminus F$. Similarly, let $X^{\text{OUT}}(v)$ be the vertex of X of maximum index (if exists) that has a path to v in $G \setminus F$. (See Figure 1).

We start by proving certain conditions that must hold for a vertex if its SCC in $G \setminus F$ intersects X.

▶ **Lemma 8.** *For any vertex $w \in V$, the SCC that contains w in $G \setminus F$ intersects X if and only if the following two conditions are satisfied.*

(i) *Both $X^{\text{IN}}(w)$ and $X^{\text{OUT}}(w)$ are defined, and*

(ii) *either $X^{\text{IN}}(w) = X^{\text{OUT}}(w)$, or $X^{\text{IN}}(w)$ appears before $X^{\text{OUT}}(w)$ on X.*

Proof. Consider any vertex $w \in V$. Let S be the SCC in $G \setminus F$ that contains w and assume S intersects X. Let w_1 and w_2 be the first and last vertices of X, respectively, that are in S. Since w and w_1 are in S there is a path from w to w_1 in $G \setminus F$. Moreover, w cannot reach a vertex that precedes w_1 in X since such a vertex will be in S as well and it will contradict the definition of w_1. Therefore, $w_1 = X^{\text{IN}}(w)$. Similarly we can prove that $w_2 = X^{\text{OUT}}(w)$. Since w_1 and w_2 are defined to be the first and last vertices from S on X, respectively, it follows that either $w_1 = w_2$, or w_1 precedes w_2 on X. Hence conditions (i) and (ii) are satisfied.

Now assume that conditions (i) and (ii) are true. The definition of $X^{\text{IN}}(\cdot)$ and $X^{\text{OUT}}(\cdot)$ implies that there is a path from $X^{\text{OUT}}(w)$ to w, and a path from w to $X^{\text{IN}}(w)$. Also, condition (ii) implies that there is a path from $X^{\text{IN}}(w)$ to $X^{\text{OUT}}(w)$. Thus w, $X^{\text{IN}}(w)$, and $X^{\text{OUT}}(w)$ are in the same SCC and it intersects X. ◀

The following lemma states the condition under which any two vertices lie in the same SCC, given that their SCCs intersect X.

▶ **Lemma 9.** *Let a, b be any two vertices in V whose SCCs intersect X. Then a and b lie in the same SCC if and only if $X^{\text{IN}}(a) = X^{\text{IN}}(b)$ and $X^{\text{OUT}}(a) = X^{\text{OUT}}(b)$.*

Algorithm 1: Binary-Search(i, j, A)

```
1 if (i = j) then
2 |   foreach v ∈ A do X^OUT(v) = x_i;
3 else
4 |   mid ← ⌈(i + j)/2⌉;
5 |   B ← Reach(x_mid, A);            /* vertices in A reachable from x_mid */
6 |   Binary-Search(i, mid-1, A\B);
7 |   Binary-Search(mid, j, B);
8 end
```

Proof. In the proof of Lemma 8, we show that if SCC of w intersects X, then $X^{\text{IN}}(w)$ and $X^{\text{OUT}}(w)$ are precisely the first and last vertices on X that lie in the SCC of w. Since SCCs forms a partition of V, vertices a and b will lie in the same SCC if and only if $X^{\text{IN}}(a) = X^{\text{IN}}(b)$ and $X^{\text{OUT}}(a) = X^{\text{OUT}}(b)$. ◀

It follows from the above two lemmas that in order to compute the SCCs in $G \setminus F$ that intersect with X, it suffices to compute $X^{\text{IN}}(\cdot)$ and $X^{\text{OUT}}(\cdot)$ for all vertices in V. It suffices to focus on computation of $X^{\text{OUT}}(\cdot)$ for all the vertices of V, since $X^{\text{IN}}(\cdot)$ can be computed in an analogous manner by just looking at graph G^R. One trivial approach to achieve this goal is to compute the set V_i consisting of all vertices reachable from each x_i by performing a BFS or DFS traversal of graph $\mathcal{G}(x_i) \setminus F$. Using this straightforward approach it takes $O(2^k nt)$ time to complete the task of computing $X^{\text{OUT}}(v)$ for every $v \in V$, while our target is to do so in $O(2^k n \log n)$ time.

Observe the nested structure underlying V_i's, that is, $V_1 \supseteq V_2 \supseteq \cdots \supseteq V_t$. Consider any vertex $x_\ell, 1 < \ell < t$. The nested structure implies for every $v \in V_\ell$ that $X^{\text{OUT}}(v)$ must be on the portion $(x_\ell, \ldots, x_t)$ of X. Similarly, it implies for every $v \in V_1 \setminus V_\ell$ that $X^{\text{OUT}}(v)$ must be on the portion $(x_1, \ldots, x_{\ell-1})$ of X. This suggests a divide and conquer approach to efficiently compute $X^{\text{OUT}}(\cdot)$. We first compute the sets V_1 and V_t in $O(2^k n)$ time each. For each $v \in V \setminus V_1$, we assign NULL to $X^{\text{OUT}}(v)$ as it is not reachable from any vertex on X; and for each $v \in V_t$ we set $X^{\text{OUT}}(v)$ to x_t. For vertices in set $V_1 \setminus V_t$, $X^{\text{OUT}}(\cdot)$ is computed by calling the function Binary-Search$(1, t-1, V_1 \setminus V_t)$. See Algorithm 1.

In order to explain the function Binary-Search, we first state an assertion that holds true for each recursive call of the function Binary-Search. We prove this assertion in the next subsection.

Assertion 1: If Binary-Search(i, j, A) is called, then A is precisely the set of those vertices $v \in V$ whose $X^{\text{OUT}}(v)$ lies on the path $(x_i, x_{i+1}, \ldots, x_j)$.

We now explain the execution of function Binary-Search(i, j, A). If $i = j$, then we assign x_i to $X^{\text{OUT}}(v)$ for each $v \in A$ as justified by Assertion 1. Let us consider the case when $i \neq j$. In this case we first compute the index $mid = \lceil (i + j)/2 \rceil$. Next we compute the set B consisting of all the vertices in A that are reachable from x_{mid}. This set is computed using the function Reach(x_{mid}, A) which is explained later in Subsection 3.2. As follows from Assertion 1, $X^{\text{OUT}}(v)$ for each vertex $v \in A$ must belong to path $(x_i, \ldots, x_j)$. Thus, $X^{\text{OUT}}(v)$ for all $v \in B$ must lie on path $(x_{mid}, \ldots, x_j)$, and $X^{\text{OUT}}(v)$ for all $v \in A \setminus B$ must lie on path $(x_i, \ldots, x_{mid\text{-}1})$. So for computing $X^{\text{OUT}}(\cdot)$ for vertices in $A \setminus B$ and B, we invoke the functions Binary-Search$(i, mid\text{-}1, A \setminus B)$ and Binary-Search(mid, j, B), respectively.

3.1 Proof of correctness of algorithm

In this section we prove that Assertion 1 holds for each call of the Binary-Search function. We also show how this assertion implies that $X^{\text{OUT}}(v)$ is correctly computed for every $v \in V$.

Let us first see how Assertion 1 implies the correctness of our algorithm. It follows from the description of the algorithm that for each $i, (1 \leq i \leq t-1)$, the function Binary-Search(i, i, A) is invoked for some $A \subseteq V$. Assertion 1 implies that A must be the set of all those vertices $v \in V$ such that $X^{\text{OUT}}(v) = x_i$. As can be seen, the algorithm in this case correctly sets $X^{\text{OUT}}(v)$ to x_i for each $v \in A$.

We now show that Assertion 1 holds true in each call of the function Binary-Search. It is easy to see that Assertion 1 holds true for the first call Binary-Search$(1, t-1, V_1 \setminus V_t)$. Consider any intermediate recursive call Binary-Search(i, j, A), where $i \neq j$. It suffices to show that if Assertion 1 holds true for this call, then it also holds true for the two recursive calls that it invokes. Thus let us assume A is the set of those vertices $v \in V$ whose $X^{\text{OUT}}(v)$ lies on the path $(x_i, x_{i+1}, \ldots, x_j)$. Recall that we compute index mid lying between i and j, and find the set B consisting of all those vertices in A that are reachable from x_{mid}. From the nested structure of the sets $V_i, V_{i+1}, \ldots, V_j$, it follows that $X^{\text{OUT}}(v)$ for all $v \in B$ must lie on path $(x_{mid}, \ldots, x_j)$, and $X^{\text{OUT}}(v)$ for all $v \in A \setminus B$ must lie on path $(x_i, \ldots, x_{mid\text{-}1})$. That is, B is precisely the set of those vertices whose $X^{\text{OUT}}(v)$ lies on the path $(x_{mid}, \ldots, x_j)$, and $A \setminus B$ is precisely the set of those vertices whose $X^{\text{OUT}}(v)$ lies on the path $(x_i, \ldots, x_{mid\text{-}1})$. Thus Assertion 1 holds true for the recursive calls Binary-Search$(i, mid\text{-}1, A \setminus B)$ and Binary-Search(mid, j, B) as well.

3.2 Implementation of function Reach

The main challenge left now is to find an efficient implementation of the function Reach which has to compute the vertices of its input set A that are reachable from a given vertex $x \in X$ in $G \setminus F$. The function Reach can be easily implemented by a standard graph traversal initiated from x in the graph $\mathcal{G}(x) \setminus F$ (recall that $\mathcal{G}(x)$ is a k-FTRS of x in G). This, however, will take $O(2^k n)$ time which is not good enough for our purpose, as the total running time of Binary-Search in this case will become $O(|X|2^k n)$. Our aim is to implement the function Reach in $O(2^k|A|)$ time. In general, for an arbitrary set A this might not be possible. This is because A might contain a vertex that is reachable from x via a single path whose vertices are not in A, therefore, the algorithm must explore edges incident to vertices that are not in A as well. However, the following lemma, that exploits Assertion 1, suggests that in our case as the call to Reach is done while running the function Binary-Search we can restrict ourselves to the set A only.

▶ **Lemma 10.** *If* Binary-Search(i, j, A) is called and $\ell \in [i, j]$, then for each path P from x_ℓ to a vertex $z \in A$ in graph in $G \setminus F$, all the vertices of P must be in the set A.

Proof. Assertion 1 implies that A is precisely the set of those vertices in V which are reachable from x_i but not reachable from x_{j+1} in $G \setminus F$. Consider any vertex $y \in P$. Observe that y is reachable from x_i by the path $X[x_i, x_\ell]::P[x_\ell, y]$. Moreover, y is not reachable from x_{j+1}, because otherwise z will also be reachable from x_{j+1}, which is not possible since $z \in A$. Thus vertex y lies in the set A. ◀

Lemma 10 and Lemma 4 imply that in order to find the vertices in A that are reachable from x_{mid}, it suffices to do traversal from x_{mid} in the graph G_A, the induced subgraph of A in $\mathcal{G}(x) \setminus F$, that has $O(2^k|A|)$ edges. Therefore, based on the above discussion, Algorithm 2 given below, is an implementation of function Reach that takes $O(2^k|A|)$ time.

Algorithm 2: Reach(x_{mid}, A)

```
1 H ← G(x_mid) \ F;
2 G_A ← (A, ∅);                                   /* an empty graph */
3 foreach v ∈ A do
4 |   foreach (y, v) ∈ In-Edges(v, H) do
5 |   |   if y ∈ A then E(G_A) = E(G_A) ∪ (y, v);
6 |   end
7 end
8 B ← Vertices reachable from x_mid obtained by a BFS or DFS traversal of graph G_A;
9 Return B;
```

The following lemma gives the analysis of running time of Binary-Search$(1, t-1, V_1 \setminus V_t)$.

▶ **Lemma 11.** *The total running time of* Binary-Search$(1, t-1, V_1 \setminus V_t)$ is $O(2^k n \log n)$.

Proof. The time complexity of Binary-Search$(1, t-1, V_1 \setminus V_t)$ is dominated by the total time taken by all invocation of function Reach. Let us consider the recursion tree associated with Binary-Search$(1, t-1, V_1 \setminus V_t)$. It can be seen that this tree will be of height $O(\log n)$. In each call of the Binary-Search, the input set A is partitioned into two disjoint sets. As a result, the input sets associated with all recursive calls at any level j in the recursion tree form a disjoint partition of $V_1 \setminus V_t$. Since the time taken by Reach is $O(2^k|A|)$, so the total time taken by all invocations of Reach at any level j is $O(2^k|V_1 \setminus V_t|)$. As there are at most $\log n$ levels in the recursion tree, the total time taken by Binary-Search$(1, t-1, V_1 \setminus V_t)$ is $O(2^k n \log n)$. ◀

We conclude with the following theorem.

▶ **Theorem 12.** *Let F be any set of at most k failed edges, and $X = \{x_1, x_2, \ldots, x_t\}$ be any path in $G \setminus F$. If we have prestored the graphs $\mathcal{G}(x)$ and $\mathcal{G}^R(x)$ for each $x \in X$, then we can compute all the SCCs of $G \setminus F$ which intersect with X in $O(2^k n \log n)$ time.*

4 Main Algorithm

In the previous section we showed that given any path P, we can compute all the SCCs intersecting P efficiently, if P is intact in $G \setminus F$. In the case that P contains ℓ failed edges from F then P is decomposed into $\ell + 1$ paths, and we can apply Theorem 12 to each of these paths separately to get the following theorem:

▶ **Theorem 13.** *Let P be any given path in G. Then there exists an $O(2^k n|P|)$ size data structure that for any arbitrary set F of at most k edges computes the SCCs of $G \setminus F$ that intersect the path P in $O((\ell+1)2^k n \log n)$ time, where ℓ $(\ell \leq k)$ is the number of edges in F that lie on P.*

Now in order to use Theorem 13 to design a fault tolerant algorithm for SCCs, we need to find a family of paths, say $\mathcal{P}$, such that for any F, each SCC of $G \setminus F$ intersects at least one path in $\mathcal{P}$. As described in the Subsection 1.1, a heavy path decomposition of DFS tree T serves as a good choice for $\mathcal{P}$. Choosing T as a DFS tree helps us because of the following reason: let P be any root-to-leaf path, and suppose we have already computed the SCCs in $G \setminus F$ intersecting P. Then each of the remaining SCCs must be contained in some subtree hanging from path P. The following lemma formally states this fact.

Algorithm 3: Compute SCC(G, F)

```
1  C ← ∅;                                          /* Collection of SCCs */
2  W ← ∅;                          /* A subset of V whose SCC have been computed */
3  P ← A heavy-path decomposition of T, where paths are sorted in the non-decreasing
     order of their depths;
4  foreach Path(a, b) ∈ P do
5  |  A ← Vertices lying in the subtree T(a);
6  |  (S_1, ..., S_t) ← SCCs intersecting Path(a, b) in G(A) \ F computed using D_{a,b};
7  |  foreach i ∈ [1, t] do
8  |  |  if (S_i ⊈ W) then Add S_i to collection C and set W = W ∪ S_i;
9  |  end
10 end
11 Return C;
```

▶ **Lemma 14.** *Let F be any set of failed edges, and $Path(a, b)$ be any path in $\mathcal{P}$. Let S be any SCC in $G \setminus F$ that intersects $Path(a, b)$ but does not intersect any ancestor path of $Path(a, b)$ in $\mathcal{P}$. Then all the vertices of S must lie in the subtree $T(a)$.*

Proof. Consider a vertex u on $Path(a, b)$ whose SCC S_u in $G \setminus F$ is not completely contained in the subtree $T(a)$. We show that S_u must contain an ancestor of a in T, thereby proving that it intersects an ancestor-path of $Path(a, b)$ in $\mathcal{P}$. Let v be any vertex in S_u that is not in the subtree $T(a)$. Let $P_{u,v}$ and $P_{v,u}$ be paths from u to v and from v to u, respectively, in $G \setminus F$. From Lemma 5 it follows that either $P_{u,v}$ or $P_{v,u}$ must pass through a common ancestor of u and v in T. Let this ancestor be z. Notice that all the vertices of $P_{u,v}$ and $P_{v,u}$ must lie in S_u. In particular, z must also lie in S_u. Moreover, since $v \notin T(a)$ and $u \in T(a)$, their common ancestor z in T is an ancestor of a. Since $z \in S_u$ and it is an ancestor of a in T, the lemma follows. ◀

Lemma 14 suggests that if we process the paths from $\mathcal{P}$ in the non-decreasing order of their depths, then in order to compute the SCCs intersecting a path $Path(a, b) \in \mathcal{P}$, it suffices to focus on the subgraph induced by the vertices in $T(a)$ only. This is because the SCCs intersecting $Path(a, b)$ that do not completely lie in $T(a)$ would have already been computed during the processing of some ancestor path of $Path(a, b)$.

We preprocess the graph G as follows. We first compute a heavy path decomposition $\mathcal{P}$ of DFS tree T. Next for each path $Path(a, b) \in \mathcal{P}$, we use Theorem 13 to construct the data structure for path $Path(a, b)$ and the subgraph of G induced by vertices in $T(a)$. We use the notation $\mathcal{D}_{a,b}$ to denote this data structure. Our algorithm for reporting SCCs in $G \setminus F$ will use the collection of these data structures associated with the paths in $\mathcal{P}$ as follows.

Let $\mathcal{C}$ denote the collection of SCCs in $G \setminus F$ initialized to $\emptyset$. We process the paths from $\mathcal{P}$ in non-decreasing order of their depths. Let $Path(a, b)$ be any path in $\mathcal{P}$ and let A be the set of vertices belonging to $T(a)$. We use the data structure $\mathcal{D}_{a,b}$ to compute SCCs of $G(A) \setminus F$ intersecting $Path(a, b)$. Let these be $S_1, \ldots, S_t$. Note that some of these SCCs might be a part of some bigger SCC computed earlier. We can detect it by keeping a set W of all vertices for which we have computed their SCCs. So if $S_i \subseteq W$, then we can discard S_i, else we add S_i to collection $\mathcal{C}$. Algorithm 3 gives the complete pseudocode of this algorithm.

Note that, in the above explanation, we only used the fact that T is a DFS tree, and $\mathcal{P}$ could have been any path decomposition of T. We now show how the fact that $\mathcal{P}$ is a heavy-path decomposition is crucial for the efficiency of our algorithm. Consider any vertex

$v \in T$. The number of times v is processed in Algorithm 3 is equal to the number of paths in $\mathcal{P}$ that start from either v or an ancestor of v. For this number to be small for each v, we choose $\mathcal{P}$ to be a heavy path decomposition of T. On applying Theorem 13, this immediately gives that the total time taken by Algorithm 3 is $O(k2^k n \log^2 n)$. In the next subsection, we do a more careful analysis to give a bound of $O(2^k n \log^2 n)$.

4.1 Analysis of time complexity of Algorithm 3

For any path $Path(a,b) \in \mathcal{P}$ and any set F of failing edges, let $\ell(a,b)$ denote the number of edges of F that lie on $Path(a,b)$. It follows from Theorem 13 that the time spent in processing $Path(a,b)$ by Algorithm 3 is $O\big((\ell(a,b)+1) \times 2^k |T(a)| \times \log n\big)$. Hence the time complexity of Algorithm 3 is of the order of

$$\sum_{Path(a,b)\in\mathcal{P}} (\ell(a,b)+1) \times 2^k |T(a)| \times \log n\,.$$

In order to calculate this we define a notation $\alpha(v, Path(a,b))$ as $\ell(a,b)+1$ if $v \in T(a)$, and 0 otherwise, for each $v \in V$ and $Path(a,b) \in \mathcal{P}$. So the time complexity of Algorithm 3 becomes

$$\begin{aligned} & 2^k \log n \times \Big(\sum_{Path(a,b)\in\mathcal{P}} (\ell(a,b)+1) \times |T(a)| \Big) \\ = {} & 2^k \log n \times \Big(\sum_{Path(a,b)\in\mathcal{P}} \sum_{v\in V} \alpha(v, Path(a,b)) \Big) \\ = {} & 2^k \log n \times \Big(\sum_{v\in V} \sum_{Path(a,b)\in\mathcal{P}} \alpha(v, Path(a,b)) \Big)\,. \end{aligned}$$

Observe that for any vertex v and $Path(a,b) \in \mathcal{P}$, $\alpha(v, Path(a,b))$ is equal to $\ell(a,b)+1$ if a is either v or an ancestor of v, otherwise it is zero. Consider any vertex $v \in V$. We now show that $\sum_{Path(a,b)\in\mathcal{P}} \alpha(v, Path(a,b))$ is at most $k + \log n$. Let P_v denote the set of those paths in $\mathcal{P}$ which starts from either v or an ancestor of v. Then $\sum_{Path(a,b)\in\mathcal{P}} \alpha(v, Path(a,b)) = \sum_{Path(a,b)\in P_v} \ell(a,b) + 1$. Note that $\sum_{Path(a,b)\in P_v} \ell(a,b)$ is at most k, and Lemma 6 implies that the number of paths in P_v is at most $\log n$. This shows that $\sum_{Path(a,b)\in\mathcal{P}} \alpha(v, Path(a,b))$ is at most $k + \log n$ which is $O(\log n)$, since $k \le \log n$.

Hence the time complexity of Algorithm 3 becomes $O(2^k n \log^2 n)$. We thus conclude with the following theorem.

▶ **Theorem 15.** *For any n-vertex directed graph G, there exists an $O(2^k n^2)$ size data structure that, given any set F of at most k failing edges, can report all the SCCs of $G \setminus F$ in $O(2^k n \log^2 n)$ time.*

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014. URL: `http://dx.doi.org/10.1109/FOCS.2014.53`, `doi:10.1109/FOCS.2014.53`.

2 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant subgraph for single source reachability: generic and optimal. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 509–518, 2016. `doi:10.1145/2897518.2897648`.

3 Aaron Bernstein and David Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *STOC'09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 101–110, New York, NY, USA, 2009. ACM. doi:http://doi.acm.org/10.1145/1536414.1536431.

4 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Multiple-edge-fault-tolerant approximate shortest-path trees. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 18:1–18:14, 2016.

5 Shiri Chechik. Fault-tolerant compact routing schemes for general graphs. *Inf. Comput.*, 222:36–44, 2013. URL: http://dx.doi.org/10.1016/j.ic.2012.10.009, doi:10.1016/j.ic.2012.10.009.

6 Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. (1 + epsilon)-approximate f-sensitive distance oracles. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1479–1496, 2017. URL: http://dx.doi.org/10.1137/1.9781611974782.96, doi:10.1137/1.9781611974782.96.

7 Shiri Chechik, Thomas Dueholm Hansen, Giuseppe F. Italiano, Jakub Lacki, and Nikos Parotsidis. Decremental single-source reachability and strongly connected components in $O(m\sqrt{n})$ total update time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 315–324, 2016.

8 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-sensitivity distance oracles and routing schemes. *Algorithmica*, 63(4):861–882, 2012. URL: http://dx.doi.org/10.1007/s00453-011-9543-0, doi:10.1007/s00453-011-9543-0.

9 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

10 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008. doi:http://dx.doi.org/10.1137/S0097539705429847.

11 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 169–178, 2011. URL: http://doi.acm.org/10.1145/1993806.1993830, doi:10.1145/1993806.1993830.

12 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *SODA'09: Proceedings of 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 506–515, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

13 Ran Duan and Seth Pettie. Connectivity oracles for failure prone graphs. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 465–474, 2010. URL: http://doi.acm.org/10.1145/1806689.1806754, doi:10.1145/1806689.1806754.

14 Ran Duan and Seth Pettie. Connectivity oracles for graphs subject to vertex failures. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 490–509, 2017.

15 Daniele Frigioni, Tobias Miller, Umberto Nanni, and Christos D. Zaroliagis. An experimental study of dynamic algorithms for transitive closure. *ACM Journal of Experimental Algorithmics*, 6:9, 2001. URL: http://doi.acm.org/10.1145/945394.945403, doi:10.1145/945394.945403.

16 Loukas Georgiadis, Giuseppe F. Italiano, and Nikos Parotsidis. Strong connectivity in directed graphs under failures, with applications. In *Proceedings of the Twenty-Eighth*

Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19, pages 1880–1899, 2017.

17 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved distance sensitivity oracles via fast single-source replacement paths. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 748–757, 2012. URL: http://dx.doi.org/10.1109/FOCS.2012.17, doi:10.1109/FOCS.2012.17.

18 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Sublinear-time decremental algorithms for single-source reachability and shortest paths on directed graphs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 674–683, 2014. URL: http://doi.acm.org/10.1145/2591796.2591869, doi:10.1145/2591796.2591869.

19 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 21–30, 2015. URL: http://doi.acm.org/10.1145/2746539.2746609, doi:10.1145/2746539.2746609.

20 Giuseppe F. Italiano. Finding paths and deleting edges in directed acyclic graphs. *Inf. Process. Lett.*, 28(1):5–11, 1988. URL: http://dx.doi.org/10.1016/0020-0190(88)90136-6, doi:10.1016/0020-0190(88)90136-6.

21 Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1131–1142, 2013.

22 Neelesh Khanna and Surender Baswana. Approximate shortest paths avoiding a failed vertex: Optimal size data structures for unweighted graphs. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, pages 513–524, 2010.

23 Jakub Lacki. Improved deterministic algorithms for decremental transitive closure and strongly connected components. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1438–1445, 2011. doi:10.1137/1.9781611973082.111.

24 Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21-23, 2015*, pages 481–490, 2015. doi:10.1145/2767386.2767408.

25 Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 779–790, 2013.

26 Mihai Patrascu and Mikkel Thorup. Planning for fast connectivity updates. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 263–271, 2007. doi:10.1109/FOCS.2007.54.

27 Liam Roditty. Decremental maintenance of strongly connected components. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1143–1150, 2013.

28 Liam Roditty and Uri Zwick. Improved dynamic reachability algorithms for directed graphs. *SIAM J. Comput.*, 37(5):1455–1471, 2008.

29 Daniel D. Sleator and Robert E. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26:362–391, 1983.

30 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972. doi:10.1137/0201010.

31 Oren Weimann and Raphael Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Transactions on Algorithms*, 9(2):14, 2013. doi:10.1145/2438645.2438646.

Preserving Distances in Very Faulty Graphs*†

Greg Bodwin[1], Fabrizio Grandoni[2], Merav Parter[3], and Virginia Vassilevska Williams[4]

1 **CSAIL, MIT, Cambridge, MA, USA**
gbodwin@mit.edu
2 **IDSIA, USI-SUPSI, Manno, Switzerland**
fabrizio@idsia.ch
3 **CSAIL, MIT, Cambridge, MA, USA**
parter@mit.edu
4 **CSAIL, MIT, Cambridge, MA, USA**
virgi@mit.edu

Abstract

Preservers and additive spanners are sparse (hence cheap to store) subgraphs that preserve the distances between given pairs of nodes exactly or with some small additive error, respectively. Since real-world networks are prone to failures, it makes sense to study fault-tolerant versions of the above structures. This turns out to be a surprisingly difficult task. For every small but arbitrary set of edge or vertex failures, the preservers and spanners need to contain *replacement paths* around the faulted set. Unfortunately, the complexity of the interaction between replacement paths blows up significantly, even from 1 to 2 faults, and the structure of optimal preservers and spanners is poorly understood. In particular, no nontrivial bounds for preservers and additive spanners are known when the number of faults is bigger than 2.

Even the answer to the following innocent question is completely unknown: what is the worst-case size of a preserver for a *single pair* of nodes in the presence of f edge faults? There are no super-linear lower bounds, nor subquadratic upper bounds for $f > 2$. In this paper we make substantial progress on this and other fundamental questions:

- We present the first truly sub-quadratic size fault-tolerant single-pair preserver in unweighted (possibly directed) graphs: for any n node graph and *any* fixed number f of faults, $\tilde{O}(fn^{2-1/2^f})$ size suffices. Our result also generalizes to the single-source (all targets) case, and can be used to build new fault-tolerant additive spanners (for all pairs).
- The size of the above single-pair preserver grows to $O(n^2)$ for increasing f. We show that this is necessary even in undirected unweighted graphs, and even if you allow for a small additive error: If you aim at size $O(n^{2-\varepsilon})$ for $\varepsilon > 0$, then the additive error has to be $\Omega(\varepsilon f)$. This surprisingly matches known upper bounds in the literature.
- For weighted graphs, we provide matching upper and lower bounds for the single pair case. Namely, the size of the preserver is $\Theta(n^2)$ for $f \geq 2$ in both directed and undirected graphs, while for $f = 1$ the size is $\Theta(n)$ in undirected graphs. For directed graphs, we have a superlinear upper bound and a matching lower bound.

Most of our lower bounds extend to the distance oracle setting, where rather than a subgraph we ask for any compact data structure.

1998 ACM Subject Classification G.2.2 Graph Theory, I.1.2 Analysis of Algorithms, B.8.1 Reliability, Testing, and Fault-Tolerance

* A full version of this paper is available at https://arxiv.org/abs/1703.10293.

† The first and fourth author were partially supported by NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, and BSF Grant BSF:2012338. The second author was partially supported by the ERC Starting Grant NEWNET 279352 and the SNSF Grant APPROXNET 200021_159697/1.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 73; pp. 73:1–73:14

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

Keywords and phrases Fault Tolerance, shortest paths, replacement paths

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.73

1 Introduction

Distance preservers and additive spanners are (sparse) subgraphs that preserve, either exactly or with some small additive error, the distances between given critical pairs P of nodes. This has been a subject of intense research in the last two decades [18, 11, 4, 3, 15, 6, 1, 34].

However, real-world networks are prone to failures. For this reason, more recently (e.g. [16, 14, 17, 32, 30, 9, 33, 8, 21, 27, 19, 28]) researchers have devoted their attention to fault-tolerant versions of the above structures, where distances are (approximately) preserved even in the presence of a few edge (or vertex) faults. For the sake of simplicity we focus here on edge faults, but many of these results generalize to the case of vertex faults where $F \subseteq V$.

▶ **Definition 1.** Given an n-node graph $G = (V, E)$ and $P \subseteq V \times V$, a subgraph $H \subseteq G$ is an f-fault tolerant (f-FT) β-additive P-pairwise spanner if

$$dist_{H \setminus F}(s,t) \leq dist_{G \setminus F}(s,t) + \beta, \quad \forall (s,t) \in P, \forall F \subseteq E, |F| \leq f.$$

If $\beta = 0$, then H is an f-FT P-pairwise preserver.

Finding sparse FT spanners/preservers turned out to be an incredibly challenging task. Despite intensive research, many simple questions have remained open, the most striking of which arguably is the following:

▶ **Question 1.** *What is the worst-case size of a preserver for a single pair (s,t) and $f \geq 1$ faults?*

Prior work [31, 32] considered the single-source $P = \{s\} \times V$ unweighted case, providing super-linear lower bounds for any f and tight upper bounds for $f = 1, 2$. However, first, there is nothing known for $f > 2$, and second, the lower bounds for the $\{s\} \times V$ case do not apply to the single pair case where much sparser preservers might exist. Prior to this work, it was conceivable that in this case $O(n)$ edges suffice for arbitrary fixed f.

Our first result is a *complete* answer to Question 1 for weighted graphs. For $f = 1$ and undirected graphs, we show that a $O(n)$ size preserver exists. Our result is achieved by proving the following interesting fact: for any replacement path $P_{s,t,e}$ protecting against a single edge fault e, there is an edge $(x, y) \in P_{s,t,e}$ such that there is no shortest path from s to x in G that includes e, and there is no shortest path from t to y in G that includes e. Therefore it is sufficient to build shortest path trees from s and to t, and then add one extra edge per possible fault e along the shortest path from s to t. With a trivial union bound, we get that any set P of node pairs can be preserved using $O(\min(n|P|, n^2))$ edges. It is natural to wonder if one can improve this union bound by doing something smarter in the construction. Surprisingly, the answer is NO: we are able to provide a matching lower bound.

▶ **Theorem 2.** *For any undirected n-node weighted graph G and any set P of p pairs of nodes, there exists a P-pairwise 1-FT preserver of size $O(\min(np, n^2))$. Furthermore, for any integer $1 \leq p \leq \binom{n}{2}$, there exists an undirected weighted graph G and a set P of p node pairs such that every 1-FT P-pairwise preserver of G contains $\Omega(\min(n|P|, n^2))$ edges.*

The lower bound part obtained by adapting the lower bound of [32] to the weighted case; this allows us to a lower bound graph whose number of edges is a function of the number of pairs.

For $f = 1$ and directed graphs, we achieve the following. Let $\mathtt{DP}(n)$ denote a tight bound for the sparsity of a pairwise distance preserver in directed weighted graphs with n nodes and $O(n)$ pairs.

▶ **Theorem 3.** *For every directed weighted n-node graph $G = (V, E)$ and for every pair of nodes $s, t \in V$, there is a 1-FT (s, t) preserver with $O(\mathtt{DP}(n))$ edges. For every n, there exists a directed weighted n-node graph $G = (V, E)$ and a node pair $s, t \in V$ such that any 1-FT (s, t) preserver for G has $\Omega(\mathtt{DP}(n))$ edges.*

Coppersmith and Elkin [18] show that $\Omega(n^{4/3}) \leq \mathtt{DP}(n) \leq O(n^{3/2})$. It is a major open question to close this gap, and we show that the no-fault n-pair distance preserver question is equivalent to the 1-fault single pair preserver question, thereby fully answering the latter question, up to resolving the major open problem for n-pair preservers.

We show that the situation dramatically changes for $f \geq 2$.

▶ **Theorem 4.** *There exists an undirected weighted graph G and a single node pair (s, t) in this graph such that every 2-FT (s, t) preserver of G requires $\Omega(n^2)$ edges.*

For unweighted graphs, we achieve several non-trivial upper and lower bounds concerning the worst-case size of (s, t) preservers and spanners. First of all, we address the following question.

▶ **Question 2.** *In unweighted graphs, is the worst-case size of an f-FT (s, t) preserver subquadratic for every constant $f \geq 2$?*

Prior work showed that the answer is YES for $f = 1, 2$ [31, 33], but nothing is known for $f \geq 3$. We show that the answer is YES. Indeed, our result is more general. First, it extends to the single-source case (i.e., $P = \{s\} \times V$) and even to a small enough set of sources (i.e., $P = S \times V$ for small $|S|$). Second, the same result holds for *any* fixed number f of vertex faults. Prior work was only able to address the simple case $f = 1$ [30]. We also remark that our preserver can be computed very efficiently in $O(fmn)$ time, and its analysis is relatively simple (e.g., compared to the cumbersome case analysis in [31]).

▶ **Theorem 5.** *For every directed or undirected unweighted graph $G = (V, E)$, integer $f \geq 1$ and $S \subseteq V$, one can construct in time $O(f\, n\, m)$ an f-FT S-sourcewise (i.e. $P = S \times V$) preserver of size $\widetilde{O}(f \cdot |S|^{1/2^f} \cdot n^{2-1/2^f})$, both in the case of edge and vertex faults.*

By standard techniques, we can exploit our S-sourcewise preserver to build an additive spanner (for all pairs): Let L be an integer parameter to be fixed later on. A vertex u is *low-degree* if it has degree less than L, otherwise it is *high-degree*. Let S be a random sample of $\Theta(\frac{n}{L} \cdot f \log n)$ vertices. Our spanner H consists of the f-VFT S-sourcewise preserver from Theorem 5 plus all the edges incident to low-degree vertices. This way we achieve:

▶ **Theorem 6.** *For every undirected unweighted graph $G = (V, E)$ and integer $f \geq 1$, there exists a randomized $\widetilde{O}(fnm)$-time construction of a $+2$-additive f-FT spanner of G of size $\widetilde{O}(f \cdot n^{2-1/(2^f+1)})$ that succeeds w.h.p.*[1].

In the above result the size of the preserver grows quickly to $O(n^2)$ for increasing f. This raises the following new question:

[1] The term w.h.p. (with high probability) here indicates a probability exceeding $1 - 1/n^c$, for an arbitrary constant $c \geq 2$. Since randomization is only used to select hitting sets, the algorithm can be derandomized.

▶ **Question 3.** *Does there exist a universal constant $\varepsilon > 0$ such that all unweighted graphs have an f-FT (s,t) preserver of size $O_f(n^{2-\varepsilon})$? What if we allow a small additive error?*

The only result with *strongly sub-quadratic* size in the above sense is an $O(f \cdot n^{4/3})$ size spanner with additive error $\Theta(f)$ [14, 8]. Can we remove or reduce the dependence of the error on f? We show that the answer is NO:

▶ **Theorem 7.** *For any two integers $q, h > 0$ and a sufficiently large n, there exists an unweighted undirected n-node graph $G = (V, E)$ and a pair $s, t \in V$ such that any $2hq$-FT $(2q-1)$-additive spanner for G for the single pair (s,t) has size $\Omega((\frac{n}{hq})^{2-2/(h+1)})$.*

▶ **Corollary 8.** *For any fixed constants $\varepsilon > 0$ and f, there exists an unweighted undirected n-node graph $G = (V, E)$ and a pair $s, t \in V$ such that any f-FT additive spanner for G for the single pair (s,t) of size $O(n^{2-\varepsilon})$ must have additive error $\Omega(\varepsilon f)$.*

Proof. This follows from Theorem 7 by choosing proper $h = \Theta(1/\varepsilon)$ and $q = \Theta(\varepsilon f)$. ◀

Hence the linear dependence in f in the additive error in [14, 8] is indeed necessary. We found this very surprising.

In Section 3 we present other related lower bounds which exploit the same basic construction plus ideas in [1, 10]: see Theorems 18, 19, and 20. In particular, we are able to achieve super-linear lower bounds for any $f \geq 2$, even if we allow for a small enough polynomial additive error n^{δ}.

So far we have focused on sparse distance preserving *subgraphs*. However, suppose that the distance estimates can be stored in a different way in memory. Data structures that store the distance information of a graph in the presence of faults are called *distance sensitivity oracles*. Distance sensitivity oracles are also intensely studied [20, 7, 38, 26, 22, 23]. Our main goal here[2] is to keep the size of the data structure as small as possible, leading to the following question.

▶ **Question 4.** *How much space do we need to preserve (exactly or with a small additive error) the distances between a given pair of nodes in the presence of f faults?*

Clearly all our preserver/spanner upper bounds extend to the oracle case, however the lower bounds might not: in principle a distance oracle can use much less space than a preserver/spanner with the same accuracy. Our main contribution here are the following incompressibility results:

▶ **Theorem 9.** *There exists an undirected weighted graph G and a single node pair (s,t) in this graph such that every 2-FT distance sensitivity oracle for the single pair (s,t) in G requires $\Omega(n^2)$ bits of space.*

Note that the optimal size for $f = 1$ is $\Theta(n)$ by simple folklore arguments, so our result completes our understanding in this setting.

We are able to achieve a super-linear lower bound for 3 faults even in the case of a small enough *polynomial additive error*: see Theorem 21 in Section 3.

[2] Other typical goals are to minimize preprocessing and query time - we will not address these.

1.1 Related Work

Fault-tolerant spanners were introduced in the geometric setting [27] (see also [28, 19]). FT-spanners with multiplicative stretch are relatively well understood: the error/sparsity for f-FT and f-VFT multiplicative spanners is (up to a small polynomial factor in f) the same as in the nonfaulty case. For f edge faults, Chechik et al. [16] showed how to construct f-FT $(2k-1)$-multiplicative spanners with size $\tilde{O}(fn^{1+\frac{1}{k}})$ for any $f, k \geq 1$. They also construct an f-VFT spanner with the same stretch and larger size. This was later improved by Dinitz and Krauthgamer [21] who showed the construction of f-VFT spanners with $2k-1$ error and $\tilde{O}\left(f^{2-\frac{1}{k}}n^{1+\frac{1}{k}}\right)$ edges.

FT additive spanners were first considered by Braunschvig, Chechik and Peleg in [14] (see also [8] for slightly improved results). They showed that FT $\Theta(f)$-additive spanners can be constructed by combining FT multiplicative spanners with (non-faulty) additive spanners. This construction, however, supports only edge faults. Parter and Peleg showed in [33] a lower bound of $\Omega(n^{1+\varepsilon_\beta})$ edges for single-source FT β-additive spanners. They also provided a construction of single-source FT-spanner with additive stretch 4 and $O(n^{4/3})$ edges that is resilient to one edge fault. The first constructions of FT-additive spanners resilient against *one vertex* fault were given in [30] and later on in [8]. Prior to our work, no construction of FT-additive spanners was known for $f \geq 2$ vertex faults.

As mentioned earlier, the computation of preservers and spanners in the non-faulty case (i.e. when $f = 0$) has been the subject of intense research in the last few decades. The current-best preservers can be found in [18, 11, 12]. Spanners are also well understood, both for multiplicative stretch [4, 25] and for additive stretch [3, 15, 6, 39, 1, 11, 15, 34, 2]. There are also a few results on "mixed" spanners with both multiplicative and additive stretch [24, 36, 6]

Distance sensitivity oracles are data structures that can answer queries about the distances in a given graph in the presence of faults. The first nontrivial construction was given by Demetrescu et al. [20] and later improved by Bernstein and Karger [7] who showed how to construct $\tilde{O}(n^2)$-space, constant query time oracles for a single edge fault for an m-edge n-node graph in $\tilde{O}(mn)$ time. The first work that considered the case of two faults (hence making the first jump from one to two) is due to Duan and Pettie in [22]. Their distance oracle has nearly optimal size of $\widetilde{O}(n^2)$ and query time of $\widetilde{O}(1)$. The case of bounded edge weights, and possibly multiple faults, is addressed in [38, 26] exploiting fast matrix multiplication techniques. The size of their oracle is super-quadratic.

The notion of FT-preservers is also closely related to the problem of constructing *replacement paths*. For a pair of vertices s and t and an edge e, the replacement path $P_{s,t,e}$ is the s-t shortest-path that avoids e[3]. The efficient computation of replacement paths is addressed, among others, in [29, 35, 38, 37]. A single-source version of the problem is studied in [26]. Single-source FT structures that preserve strong connectivity have been studied in [5].

1.2 Preliminaries and Notation

Assume throughout that all shortest paths ties are broken in a consistent manner. For every $s, t \in V$ and a subgraph $G' \subseteq G$, let $\pi_{G'}(s,t)$ be the (unique) u-v shortest path in G' (i.e., it is unique under breaking ties). If there is no path between s and t in G', we define

[3] Replacement paths were originally defined for the single edge fault case, but later on extended to the case of multiple faults as well.

$\pi_{G'}(s,t) = \emptyset$. When $G' = G$, we simply write $\pi(u,v)$. For any path P containing nodes u, v, let $P[u \rightsquigarrow v]$ be the subpath of P between u and v. For $s, t \in V$ and $F \subseteq E$, we let $P_{s,t,F} = \pi_{G \setminus F}(s,t)$ be the s-t shortest-path in $G \setminus F$. We call such paths *replacement paths*. When $F = \{e\}$, we simply write $P_{s,t,e}$. By m we denote the number of edges in the graph currently being considered.

The structure of the paper is as follows. In Sec. 2, we describe an efficient construction for FT-preservers and additive spanners with a subquadratic number of edges. Then, in Sec. 3, we provide several lower bound constructions for a single s-t pair, both for the exact and for the additive stretch case. All the proofs which are omitted due to lack of space appear in the full version of the paper (see [13]).

2 Efficient Construction of FT-Preservers and Spanners

In this section we prove Theorem 5. We next focus on the directed case, the undirected one being analogous and simpler. We begin by recapping the currently-known approaches for handling many faults, and we explain why these approaches fail to achieve interesting space/construction time bounds for large f.

The limits of previous approaches. A known approach for handling many faults is by *random sampling* of subgraphs, as introduced by Weimann and Yuster [38] in the setting of distance sensitivity oracles, and later on applied by Dinitz and Kraughgamer [21] in the setting of fault tolerant spanners. The high level idea is to generate multiple subgraphs $G_1, \ldots, G_r$ by removing each edge/vertex independently with sufficiently large probability p; intuitively, each G_i simultaneously captures many possible fault sets of size f. One can show that, for a sufficiently small parameter L and for any given (*short*) replacement path $P_{s,t,F}$ of length at most L (avoiding faults F), w.h.p. in at least one G_i the path $P_{s,t,F}$ is still present while all edges/vertices in F are deleted. Thus, if we compute a (non-faulty) preserver $H_i \subseteq G_i$ for each i, then the graph $H = \bigcup_i H_i$ will contain every short replacement path. For the remaining (*long*) replacement paths, Weimann and Yuster use a random decomposition into short subpaths. Unfortunately, any combination of the parameters p, r, L leads to a quadratic (or larger) space usage.

Another way to handle multiple faults is by extending the approach in [32, 33, 30] that works for $f \in \{1, 2\}$. A useful trick used in those papers (inspired by prior work in [35, 37]) is as follows: suppose $f = 1$, and fix a target node t. Consider the shortest path $\pi(s,t)$. It is sufficient to take the *last* edge of each replacement path $P_{s,t,e}$ and charge it to the node t; the rest of the path is then charged to other nodes by an inductive argument. Hence, one only needs to bound the number of *new-ending* paths – those that end in an edge that is not already in $\pi(s,t)$. In the case $f = 1$, these new-ending paths have a nice structure: they diverge from $\pi(s,t)$ at some vertex b (*divergence point*) above the failing edge/vertex and collide again with $\pi(s,t)$ only at the terminal t; the subpath connecting b and t on the replacement path is called its *detour*. One can divide the s-t replacement paths into two groups: short (resp., long) paths are those whose detour has length at most (resp., at least) $\sqrt{n}$. It is then straightforward enough to show that each category of path contributes only $\widetilde{O}(n^{1/2})$ edges entering t, and so (collecting these last edges over all nodes in the graph) the output subgraph has $\widetilde{O}(n^{3/2})$ edges in total. Generalizing this to the case of multiple faults is non-trivial already for the case of $f = 2$. The main obstacle here stems from a lack of structural understanding of replacement paths for multiple faults: in particular, any given divergence point $b \in \pi(s,t)$ can now be associated with many new-ending paths and not

only one! In the only known positive solution for $f = 2$ [31], the approach works only for edge faults and is based on an extensive case analysis whose extension to larger f is beyond reasonable reach. Thus, in the absence of new structural understanding, further progress seems very difficult.

A second source of difficulties is related to the *running time* of the construction. A priori, it seems that constructing a preserver H should require computing all replacement paths $P_{s,t,F}$, which leads to a construction time that scales *exponentially* in f. In particular, by deciding to omit an edge e from the preserver H, we must somehow check that this edge does not appear on *any* of the replacement paths $P_{s,t,F}$ (possibly, without computing these replacement paths explicitly).

Our basic approach. The basic idea behind our algorithm is as follows. Similar to [32, 33, 30], we focus on each target node t, and define a set E_t of edges incident to t to be added to our preserver. Intuitively, these are the last edges of new-ending paths as described before. The construction of E_t, however, deviates substantially from prior work. Let us focus on the simpler case of edge deletions. The set E_t is constructed recursively, according to parameter f. Initially we consider the shortest path tree T from the source set S to t, and add to E_t the edges of T incident to t (at most $|S|$ many). Consider any new-ending replacement path P for t. By the previous discussion, this path has to leave T at some node b and it meets T again only at t: let D be the subpath of P between b and t (the *detour* of P). Note that D is edge-disjoint from T, i.e. it is contained in the graph $G' = G \setminus E(T)$. Therefore, it would be sufficient to compute recursively the set E'_t of final edges of new-ending replacement paths for t in the graph G' with source set S' given by the possible divergence points b and w.r.t. $f - 1$ faults (recall that one fault must be in $E(T)$, hence we avoid that anyway in G'). This set E'_t can then be added to E_t.

The problem with this approach is that S' can contain $\Omega(n)$ many divergence points (hence E_t $\Omega(n)$ many edges), leading to a trivial $\Omega(n^2)$ size preserver. In order to circumvent this problem, we classify the divergence points b in two categories. Consider first the nodes b at distance at most L from t along T, for some parameter L. There are only $O(|S|L)$ many such nodes S^{short}, which is sublinear for $|S|$ and L small enough. Therefore we can safely add S^{short} to S'. For the remaining divergence points b, we observe that the corresponding detour D must have length at least L: therefore by sampling $\tilde{O}(n/L)$ nodes S^{long} we hit all such detours w.h.p. Suppose that $\sigma \in S^{long}$ hits detour D. Then the portion of D from σ to t also contains the final edge of D to be added to E_t. In other terms, it is sufficient to add S^{long} (which has sublinear size for polynomially large L) to S' to cover all the detours of nodes b of the second type. Altogether, in the recursive call we need to handle one less fault w.r.t. a larger (but sublinear) set of sources S'. Our approach has several benefits:

- It leads to a subquadratic size for any f (for a proper choice of the parameters);
- It leads to a very fast algorithm. In fact, for each target t we only need to compute a BFS tree in f different graphs, leading to an $O(fnm)$ running time;
- Our analysis is very simple, much simpler than in [31] for the case $f = 2$;
- It can be easily extended to the case of vertex faults.

Algorithm for Edge Faults. Let us start with the edge faults case. The algorithm constructs a set E_t of edges incident to each target node $t \in V$. The final preserver is simply the union $H = \bigcup_{t \in V} E_t$ of these edges. We next describe the construction of each E_t (see also Alg. 1). The computation proceeds in rounds $i = 0, \ldots, f$. At the beginning of round i we are given a subgraph G_i (with $G_0 = G$) and a set of sources S_i (with $S_0 = S$).

Algorithm 1 Construction of E_t in our f-FT S-Sourcewise Preserver Algorithm.

procedure ComputeSourcewiseFT(t, S, f, G)
Input: A graph G with a source set S and terminal t, number of faults f.
Output: Edges E_t incident to t in an f-FT S-sourcewise preserver H.
Set $G_0 = G$, $S_0 = S$, $E_t = \emptyset$.
  for $i \in \{0, \ldots, f\}$ **do**
    Compute the partial BFS tree $T_i = \bigcup_{s \in S_i} \pi_{G_i}(s, t)$.
    $E_t = E_t \cup \{\texttt{LastE}(\pi_{T_i}(s, t)) \mid s \in S_i\}$.
    Set distance threshold $d_i = \sqrt{n/|S_i| \cdot f \log n}$.
    Let $S_i^{short} = \{v \in V(T_i) \mid \texttt{dist}_{T_i}(v, t) \leq d_i\}$.
    Sample a collection $S_i^{long} \subseteq V(G_i)$ of $\Theta(n/d_i \cdot f \log n)$ vertices.
    Set $S_{i+1} = S_i^{short} \cup S_i^{long}$ and $G_{i+1} = G_i \setminus E(T_i)$.

We compute a partial BFS tree $T_i = \bigcup_{s \in S_i} \pi_{G_i}(s, t)$[4] from S_i to t, and add to E_t (which is initially empty) the edges $\{\texttt{LastE}(\pi_{T_i}(s, t)) \mid s \in S_i\}$ of this tree incident to t. Here, for a path π where one endpoint is the considered target node t, we denote by $\texttt{LastE}(\pi)$ the edge of π incident to t. The source set S_{i+1} is given by $S_i^{short} \cup S_i^{long}$. Here $S_i^{short} = \{v \in V(T_i) \mid \texttt{dist}_{T_i}(v, t) \leq d_i\}$ is the set of nodes at distance at most $d_i = \sqrt{n/|S_i| \cdot f \log n}$ from t, while S_i^{long} is a random sample of $\Theta(n/d_i \cdot f \log n)$ vertices. The graph G_{i+1} is obtained from G_i be removing the edges $E(T_i)$[5].

Adaptation for Vertex Faults. The only change in the algorithm is in the definition of the graph G_i inside the procedure to compute E_t. We cannot allow ourselves to remove all the vertices of the tree T_i from G_i and hence a more subtle definition is required. To define G_{i+1}, we first remove from G_i: (1) all edges of $S_i^{short} \times S_i^{short}$, (2) the edges of $E(T_i)$, and (3) the vertices of $V(T_i) \setminus S_i^{short}$. Finally, we delete all remaining edges incident to S_i^{short} which are directed towards any one of these vertices (i.e., the incoming degree of the S_i^{short} vertices in G_{i+1} is zero).

Analysis. We now analyze our algorithm. Since for each vertex t, we compute f (partial) BFS trees, we get trivially:

▶ **Lemma 10** (Running Time). *The subgraph H is computed within $O(f\, n\, m)$ time.*

We proceed with bounding the size of H.

▶ **Lemma 11** (Size Analysis). $|E_t| = \widetilde{O}(|S|^{1/2^f} \cdot (fn)^{1-1/2^f})$ *for every $t \in V$, hence* $|E(H)| = \widetilde{O}(f|S|^{1/2^f} n^{2-1/2^f})$.

Proof. Since the number of edges collected at the end each round i is bounded by the number of sources S_i, it is sufficient to bound $|S_i|$ for all i. Observe that for every $i \in \{0, \ldots, f-1\}$,

$$|S_{i+1}| \leq |S_i^{long}| + |S_i^{short}| \leq d_i \cdot |S_i| + \Theta(n/d_i \cdot f \log n) = \Theta(d_i \cdot |S_i|).$$

By resolving this recurrence starting with $|S_0| = |S|$ one obtains

$$|S_i| = O(|S|^{1/2^i} (fn \log n)^{1-1/2^i}).$$

The claim follows by summing over $i \in \{0, \ldots, f\}$. ◀

[4] If $\pi_{G_i}(s, t)$ does not exist, recall that we define it as an empty set of edges.
[5] Note that for $f = 1$, the algorithm has some similarity to the replacement path computation of [35]. Yet, there was no prior extension of this idea for $f \geq 2$.

We next show that the algorithm is correct. We focus on the vertex fault case, the edge fault case being similar and simpler. Let us define, for $t \in V$ and $i \in \{0, \ldots, f\}$,

$$\mathcal{P}_{t,i} = \{\pi_{G_i \setminus F}(s,t) \mid s \in S_i,\ F \subseteq V(G_i),\ |F| \leq f - i\}.$$

▶ **Lemma 12.** *For every $t \in V$ and $i \in \{0, \ldots, f\}$, it holds that*

$$\texttt{LastE}(\pi) \in E_t \quad \textit{for every} \quad \pi \in \mathcal{P}_{t,i}.$$

Proof. We prove the claim by decreasing induction on $i \in \{f, \ldots, 0\}$. For the base case $i = f$, $\mathcal{P}_{t,f} = \{\pi_{G_f}(s,t) \mid s \in S_f\}$. Since we add precisely the last edges of these paths to the set E_t, the claim holds. Assume that the lemma holds for rounds $f, f-1, \ldots, i+1$ and consider round i. For every $\pi_{G_i \setminus F}(s,t) \in \mathcal{P}_{t,i}$, let $P'_{s,t,F} = \pi_{G_i \setminus F}(s,t)$. [6] Consider the partial BFS tree $T_i = \bigcup_{s \in S_i} \pi_{G_i}(s,t)$ rooted at t. Note that all (interesting) replacement paths $P'_{s,t,F}$ in G_i have at least one failing vertex $v \in F \cap V(T_i)$ as otherwise $P'_{s,t,F} = \pi_{G_i}(s,t)$.

We next partition the replacement paths $\pi \in \mathcal{P}_{t,i}$ into two types depending on their last edge $\texttt{LastE}(\pi)$. The first class contains all paths whose last edge is in T_i. The second class contains the remaining replacement paths, which end with an edge that is not in T_i. We call this second class of paths *new-ending* replacement paths. Observe that the first class is taken care of, since we add all edges incident to t in T_i. Hence it remains to prove the lemma for the set of new-ending paths.

For every new-ending path $P'_{s,t,F}$, let $b_{s,t,F}$ be the last vertex on $P'_{s,t,F}$ that is in $V(T_i) \setminus \{t\}$. We call the vertex $b_{s,t,F}$ the *last divergence point* of the new-ending replacement path. Note that the detour $D_{s,t,F} = P'_{s,t,F}[b_{s,t,F} \rightsquigarrow t]$ is vertex disjoint with the tree T_i except for the vertices $b_{s,t,F}$ and t. From now on, since we only wish to collect last edges, we may restrict our attention to this detour subpath. That is, since $\texttt{LastE}(D_{s,t,F}) = \texttt{LastE}(P'_{s,t,F})$, it is sufficient to show that $\texttt{LastE}(D_{s,t,F}) \in E_t$.

Our approach is based on dividing the set of new-ending paths in $\mathcal{P}_{t,i}$ into two classes based on the position of their last divergence point $b_{s,t,F}$. The first class $\mathcal{P}_{short}$ consists of new-ending paths in $\mathcal{P}_{t,i}$ whose last divergence point is at distance at most $d_i = \sqrt{n/|S_i|} \cdot f \log n$ from t on T_i. In other words, this class contains all new-ending paths whose last divergence point is in the set S_i^{short}. We now claim the following.

▶ **Claim 13.** *For every $P'_{s,t,F} \in \mathcal{P}_{short}$, the detour $D_{s,t,F}$ is in $\mathcal{P}_{t,i+1}$.*

Proof. Since $D_{s,t,F}$ is a subpath of the replacement path $P'_{s,t,F}$, $D_{s,t,f}$ is the shortest path between $b_{s,t,F}$ and t in $G_i \setminus F$. Recall that $D_{s,t,F}$ is vertex disjoint with $V(T_i) \setminus \{b_{s,t,F}, t\}$.

Since $b_{s,t,F}$ is the last divergence point of $P'_{s,t,F}$ with T_i, the detour $D_{s,t,F}$ starts from a vertex $b_{s,t,F} \in S_i^{short}$ and does not pass through any other vertex in $V(T_i) \setminus \{t\}$. Recall that in the construction of G_{i+1} we delete from G_i the edges directed *towards* S_i^{short}. In particular, the outgoing edge connecting $b_{s,t,F}$ to its neighbor x on $D_{s,t,F}[b_{s,t,F} \rightsquigarrow t]$ remains (i.e., this vertex x is not in $V(T_i) \setminus \{t\}$), this implies that the detour $D_{s,t,F}$ exists in G_{i+1}. In particular, note that the vertex $b_{s,t,F}$ cannot be a neighbor of t in T_i. Indeed, if $(b_{s,t,F}, t)$ were an edge in T_i, then we can replace the portion of the detour path between $b_{s,t,F}$ and t by this edge, getting a contradiction to the fact that $P'_{s,t,F}$ is a new-ending path[7].

Next, observe that at least one of the failing vertices in F occurs on the subpath $\pi_{G_i}[b_{s,t,F}, t]$, let this vertex be $v \in F$. Since $v \in S_i^{short}$, all the edges incident to v are

[6] We denote these replacement paths as $P'_{s,t,F}$ as they are computed in G_i and not in G.

[7] For the edge fault case, the argument is much simpler: by removing $E(T_i)$ from G_i, we avoid at least one of the failing edges in G_{i+1}.

directed away from v in G_{i+1} and hence the paths going out from the source $b_{s,t,F}$ in G_{i+1} cannot pass through v. Letting $F' = F \setminus V(T_i)$, it holds that (1) $|F'| \leq f - i - 1$ and (2) since the shortest path ties are decided in a consistent manner and by definition of G_{i+1}, it holds that $D_{s,t,F} = \pi_{G_{i+1} \setminus F'}(b_{s,t,F}, t)$. As $b_{s,t,F} \in S_i^{short}$, it holds that $D_{s,t,F} \in \mathcal{P}_{t,i+1}$. ◀

Hence by the inductive hypothesis for $i+1$, $\texttt{LastE}(P'_{s,t,F})$ is in E_t for every $P'_{s,t,F} \in \mathcal{P}_{short}$. We now turn to consider the second class of paths $\mathcal{P}_{long}$ which contains all remaining new-ending paths; i.e., those paths whose last divergence point is at distance at least d_i from t on T_i. Note that the detour $D_{s,t,F} = P'_{s,t,F}[b_{s,t,F} \rightsquigarrow t]$ of these paths is long – i.e., its length is at least d_i. For convenience, we will consider the internal part $D'_{s,t,F} = D_{s,t,F} \setminus \{b_{s,t,F}, t\}$ of these detours, so that the first and last vertices of these detours are not on T_i.

We exploit the lengths of these detours $D'_{s,t,F}$ and claim that w.h.p, the set S_i^{long} is a hitting set for these detours. This indeed holds by simple union bound overall possible $O(n^{f+2})$ detours. For every $P'_{s,t,F} \in \mathcal{P}_{long}$, let $w_{s,t,F} \in V(D'_{s,t,F}) \cap S_i^{long}$. (By the hitting set property, w.h.p., $w_{s,t,F}$ is well defined for each long detour). Let $W_{s,t,F} = P'_{s,t,F}[w_{s,t,F}, t]$ be the suffix of the path $P'_{s,t,F}$ starting at a vertex from the hitting set $w_{s,t,F} \in S_i^{long}$. Since $\texttt{LastE}(P'_{s,t,F}) = \texttt{LastE}(W_{s,t,F})$, it is sufficient to show that $\texttt{LastE}(W_{s,t,F})$ is in E_t.

▶ **Claim 14.** *For every $P'_{s,t,F} \in \mathcal{P}_{long}$, it holds that $W_{s,t,F} \in \mathcal{P}_{t,i+1}$.*

Proof. Clearly, $W_{s,t,f}$ is the shortest path between $w_{s,t,F}$ and t in $G_i \setminus F$. Since $W_{s,t,F} \subseteq D'_{s,t,F}$ is vertex disjoint with $V(T_i)$, it holds that $W_{s,t,F} = \pi_{G_{i+1} \setminus F'}(w_{s,t,F}, t)$ for $F' = F \setminus V(T_i)$. Note that since at least one fault occurred on T_i, we have that $|F'| \leq f - i - 1$. As $w_{s,t,F} \in S_i^{long}$, it holds that $W_{s,t,F} \in \mathcal{P}_{t,i+1}$. The lemma follows. ◀

By applying the claim for $i = 0$, we get that $\texttt{LastE}(P'_{s,t,F})$ is in E_t as required for every $P'_{s,t,F} \in \mathcal{P}_{long}$. This completes the proof. ◀

▶ **Lemma 15.** *(Correctness) H is an f-FT S-sourcewise preserver.*

Proof. By using Lemma 12 with $i = 0$, we get that for every $t \in V$, $s \in S$ and $F \subseteq V$, $|F| \leq f$, $\texttt{LastE}(P_{s,t,F}) \in E_t$ (and hence also $\texttt{LastE}(P_{s,t,F}) \in H$). It remains to show that taking the last edge of each replacement path $P_{s,t,F}$ is sufficient. The base case is for paths of length 1, where we have clearly kept the entire path in our preserver. Then, assuming the hypothesis holds for paths up to length $k-1$, consider a path $P_{s,t,F}$ of length k. Let $\texttt{LastE}(P_{s,t,F}) = (u, t)$. Then since we break ties in a consistent manner, $P_{s,t,F} = P_{s,u,F} \circ \texttt{LastE}(P_{s,t,e})$. By the inductive hypothesis $P_{s,u,F}$ is in H, and since we included the last edge, $P_{s,t,F}$ is also in H. The claim follows. ◀

Theorem 5 now immediately follows from Lemmas 10, 11, and 15.

3 Lower Bounds for FT Preservers and Additive Spanners

In this section, we provide the first non-trivial lower bounds for preservers and additive spanners for a single pair s-t.

We start by proving Theorem 7. The main building block in our lower bound is the construction of an (undirected unweighted) tree T^h, where h is a positive integer parameter related to the desired number of faults f. Tree T^h is taken from [31] with mild technical adaptations. Let d be a *size* parameter which is used to obtain the desired number n of nodes. It is convenient to interpret this tree as rooted at a specific node (though edges in this construction are undirected). We next let $rt(T^h)$ and $L(T^h)$ be the root and leaf set

of T^h, respectively. We also let $\ell(h)$ and $n(h)$ be the height and number of nodes of T^h, respectively.

Tree T^h is constructed recursively as follows. The base case is given by T^0 which consists of a single isolated root node $rt(T^0)$. Note that $\ell(0) = 0$ and $n(0) = 1$. In order to construct T^h, we first create d copies $T_0^{h-1}, \ldots, T_{d-1}^{h-1}$ of T^{h-1}. Then we add a path $v_0, \ldots, v_{d-1}$ of length $d-1$ (consisting of new nodes), and choose $rt(T^h) = v_0$. Finally, we connect v_j to $rt(T_j^{h-1})$ with a path (whose internal nodes are new) of length $(d-j) \cdot (\ell(h-1)+3)$. Next lemma illustrates the crucial properties of T^h.

▶ **Lemma 16.** *The tree T^h satisfies the following properties:*

1. $n(h) \leq \frac{3}{2}(h+1)(d+1)^{h+1}$
2. $|L(T^h)| = d^h$
3. *For every $\ell \in L(T^h)$, there exists $F_\ell \subseteq E(T)$, $|F_\ell| = h$, such that $\texttt{dist}_{T^h \setminus F_\ell}(s, \ell) \leq \texttt{dist}_{T^h \setminus F_\ell}(s, \ell') + 2$ for every $\ell' \in L(T^h) \setminus \{\ell'\}$.*

We next construct a graph S^h as follows. We create two copies T_s and T_t of T^h. We add to S^h the complete bipartite graph with sides $L(T_s)$ and $L(T_t)$, which we will call the *bipartite core* B of S^h. Observe that $|L(T_s)| = |L(T_t)| = d^h$, and hence B contains d^{2h} edges. We will call $s = sr(S^h) = rt(T_s)$ the source of S^h, and $t = tg(S^h) = rt(T_t)$ its target.

▶ **Lemma 17.** *Every $2h$-FT (s,t) preserver (and 1-additive (s,t) spanner) H for S^h must contain each edge $e = (\ell_s, \ell_t) \in B$.*

Proof. Assume that $e = (\ell_s, \ell_t) \notin H$ and consider the case where F_{ℓ_s} fails in T_s and F_{ℓ_t} fails in T_t. Let $G' := S^h \setminus (F_{\ell_s} \cup F_{\ell_t})$, and d_s (resp., d_t) be the distance from s to ℓ_s (resp., from ℓ_t to t) in G'. By Lemma 16.3 the shortest s-t path in G' passes through e and has length $d_s + 1 + d_t$. By the same lemma, any path in G', hence in $H' := H \setminus (F_{\ell_s} \cup F_{\ell_t})$, that does not pass through ℓ_s (resp., ℓ_t) must have length at least $(d_s + 2) + 1 + d_t$ (resp., $d_s + 1 + (d_t + 2)$). On the other hand, any path in H' that passes through ℓ_s and ℓ_t must use at least 3 edges of B, hence having length at least $d_s + 3 + d_t$. ◀

Our lower bound graph S_q^h is obtained by taking q copies $S_1, \ldots, S_q$ of graph S^h with $d = (\frac{n}{3q(h+1)} - 1)^{\frac{1}{h+1}}$, and chaining them with edges $(tg(S_i), sr(S_{i+1}))$, for $i = 1, \ldots, q-1$. We let $s = sr(S_1)$ and $t = tg(S_q)$.

Proof of Theorem 7. Consider S_q^h. By Lemma 16.1–2 this graph contains at most n nodes, and the bipartite core of each S_i contains $d^{2h} = \Omega((\frac{n}{qh})^{2-2/(h+1)})$ edges.

Finally, we show that any $(2q-1)$-additive (s,t) spanner needs to contain all the edges of at least one such bipartite core. Let us assume this does not happen, and let e_i be a missing edge in the bipartite core of S_i for each i. Observe that each s-t shortest path has to cross $sr(S_i)$ and $tg(S_i)$ for all i. Therefore, it is sufficient to choose $2h$ faulty edges corresponding to each e_i as in Lemma 17. This introduces an additive stretch of 2 in the distance between s and t for each e_i, leading to a total additive stretch of at least $2q$. ◀

The same construction can also be extended to the setting of $(2h)$-FT $S \times T$ preservers. To do that, we make *parallel* copies of the S^h graph.

▶ **Theorem 18.** *For every positive integer f, there exists a graph $G = (V, E)$ and subsets $S, T \subseteq V$, such that every $(2f)$-FT 1-additive $S \times T$ spanner (hence $S \times T$ preserver) of G has size $\Omega(|S|^{1/(f+1)} \cdot |T|^{1/(f+1)} \cdot (n/f)^{2-2/(f+1)})$.*

Improving over the Bipartite Core. The proof above only gives the trivial lower bound of $\Omega(n)$ for the case of two faults (using $h = q = 1$). We can strengthen the proof in this special case to show instead that $\Omega(n^{1+\varepsilon})$ edges are needed, and indeed this even holds in the presence of a *polynomial additive stretch*:

▶ **Theorem 19.** *A 2-FT distance preserver of a single (s,t) pair in an undirected unweighted graph needs $\Omega(n^{11/10-o(1)})$ edges.*

▶ **Theorem 20.** *There are absolute constants $\varepsilon, \delta > 0$ such that any $+n^{\delta}$-additive 2-FT preserver for a single (s,t) pair in an undirected unweighted graph needs $\Omega(n^{1+\varepsilon})$ edges.*

Finally, by tolerating one additional fault, we can obtain a strong incompressibility result:

▶ **Theorem 21.** *There are absolute constants $\varepsilon, \delta > 0$ such that any $+n^{\delta}$-additive 3-FT distance sensitivity oracle for a single (s,t) pair in an undirected unweighted graph uses $\Omega(n^{1+\varepsilon})$ bits of space.*

The proofs of Theorems 19, 20 and 21 are similar in spirit. The key observation is that the structure of T_s, T_t allows us to use our faults to select leaves ℓ_s, ℓ_t and enforce that a shortest ℓ_s-ℓ_t path is kept in the graph. When we use a bipartite core between the leaves of T_s and T_t, this "shortest path" is simply an edge, so the quality of our lower bound is equal to the product of the leaves in T_s and T_t. However, sometimes a better graph can be used instead. In the case $h = 1$, we can use a nontrivial lower bound graph against (non-faulty) subset distance preservers (from [10]), which improves the cost per leaf pair from 1 edge to roughly $n^{11/10}$ edges, yielding Theorem 19. Alternatively, we can use a nontrivial lower bound graph against $+n^{\delta}$ spanners (from [1]), which implies Theorem 20. The proof of Theorem 21 is similar in spirit, but requires an additional trick in which *unbalanced trees* are used: we take T_s as a copy of T^1 and T_t as a copy of T^2, and this improved number of leaf-pairs is enough to push the incompressibility argument through.

4 Open Problems

There are lots of open ends to be closed. Perhaps the main open problem is to resolve the current gap for f-FT single-source preservers. Since the lower bound of $\Omega(n^{2-1/(f+1)})$ edges given in [31] has been shown to be tight for $f \in [1,2]$, it is reasonable to believe that this is the right bound for $f \geq 3$. Another interesting open question involves lower bounds for FT additive spanners. Our lower-bounds are super linear only for $f \geq 2$. The following basic question is still open though: is there a lower bound of $\Omega(n^{3/2+\epsilon})$ edges for some $\epsilon \in (0,1]$ for 2-additive spanners with *one* fault? Whereas our lower bound machinery can be adapted to provide non trivial bounds for different types of f-FT P-preservers (e.g., $P = \{s,t\}, P = S \times T$, etc.), our upper bounds technique for general $f \geq 2$ is still limited to the sourcewise setting. Specifically, it is not clear how to construct an f-FT $S \times S$ preserver other than taking a (perhaps wasteful) f-FT S-sourcewise preserver. As suggested by our lower bounds, these questions are interesting already for a single pair.

References

1 A. Abboud and G. Bodwin. The 4/3 additive spanner exponent is tight. In *STOC*, pages 351–361, 2016.

2 Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. In *SODA*, pages 568–576, 2017.

3 D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.

4 I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

5 S. Baswana, K. Choudhary, and L. Roditty. Fault tolerant subgraph for single source reachability: generic and optimal. In *STOC*, pages 509–518, 2016.

6 S. Baswana, K. Telikepalli, K. Mehlhorn, and S. Pettie. New constructions of (alpha, beta)-spanners and purely additive spanners. In *SODA*, pages 672–681, 2005.

7 Aaron Bernstein and David R. Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *STOC*, pages 101–110, 2009.

8 D. Bilò, F. Grandoni, L. Gualà, S. Leucci, and G. Proietti. Improved purely additive fault-tolerant spanners. In *ESA*, pages 167–178. Springer, 2015.

9 D. Bilò, L. Gualà, S. Leucci, and G. Proietti. Fault-tolerant approximate shortest-path trees. In *ESA*, pages 137–148. Springer, 2014.

10 G. Bodwin. Linear size distance preservers. In *SODA*, 2017.

11 G. Bodwin and V. Vassilevska Williams. Better distance preservers and additive spanners. In *SODA*, pages 855–872, 2016.

12 Greg Bodwin. Linear size distance preservers. In *SODA*, pages 600–615, 2017.

13 Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. *CoRR*, abs/1703.10293, 2017. URL: `http://arxiv.org/abs/1703.10293`.

14 G. Braunschvig, S. Chechik, D. Peleg, and A. Sealfon. Fault tolerant additive and (μ, α)-spanners. *Theor. Comput. Sci.*, 580:94–100, 2015.

15 S. Chechik. New additive spanners. In *SODA*, pages 498–512, 2013.

16 S. Chechik, M. Langberg, D. Peleg, and L. Roditty. Fault-tolerant spanners for general graphs. In *STOC*, pages 435–444, 2009.

17 S. Chechik and D. Peleg. Rigid and competitive fault tolerance for logical information structures in networks. In *Electrical and Electronics Engineers in Israel (IEEEI), 2010 IEEE 26th Convention of*, pages 000024–000025. IEEE, 2010.

18 D. Coppersmith and M. Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006.

19 A. Czumaj and H. Zhao. Fault-tolerant geometric spanners. *Discrete & Computational Geometry*, 32(2):207–230, 2004.

20 C. Demetrescu, M. Thorup, R. A. Chowdhury, and V. Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM Journal on Computing*, 37(5):1299–1318, 2008.

21 M. Dinitz and R. Krauthgamer. Fault-tolerant spanners: better and simpler. In *PODC*, pages 169–178, 2011.

22 R. Duan and S. Pettie. Dual-failure distance and connectivity oracles. In *SODA*, pages 506–515, 2009.

23 Ran Duan and Seth Pettie. Connectivity oracles for graphs subject to vertex failures. In *SODA*, pages 490–509, 2017.

24 M. Elkin and D. Peleg. (1+epsilon, beta)-spanner constructions for general graphs. *SIAM J. Comput.*, 33(3):608–631, 2004.

25 P. Erdös. Extremal problems in graph theory. *Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963)*, pages 29–36, 1963.

26 F. Grandoni and V. Vassilevska Williams. Improved distance sensitivity oracles via fast single-source replacement paths. In *FOCS*, pages 748–757, 2012.

27 C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32(1):144–156, 2002.

28 T. Lukovszki. New results on fault tolerant geometric spanners. In *Algorithms and Data Structures*, pages 193–204. Springer, 1999.

29 K. Malik, A. K. Mittal, and S. K. Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8(4):223–227, 1989.

30 M. Parter. Vertex fault tolerant additive spanners. In *Distributed Computing*, pages 167–181. Springer, 2014.

31 M. Parter. Dual failure resilient BFS structure. In *PODC*, pages 481–490, 2015.

32 M. Parter and D. Peleg. Sparse fault-tolerant BFS trees. In *ESA*, pages 779–790, 2013.

33 M. Parter and D. Peleg. Fault tolerant approximate BFS structures. In *SODA*, pages 1073–1092, 2014.

34 S. Pettie. Low distortion spanners. *ACM Transactions on Algorithms*, 6(1), 2009.

35 L. Roditty and U. Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Transactions on Algorithms*, 8(4):33, 2012.

36 M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. In *SODA*, pages 802–809, 2006.

37 V. Vassilevska Williams. Faster replacement paths. In *SODA*, pages 1337–1346. SIAM, 2011.

38 O. Weimann and R. Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Transactions on Algorithms*, 9(2):14, 2013.

39 D. P. Woodruff. Additive spanners in nearly quadratic time. In *ICALP*, pages 463–474, 2010.

All-Pairs 2-Reachability in $\mathcal{O}(n^\omega \log n)$ Time*

Loukas Georgiadis[1], Daniel Graf[2], Giuseppe F. Italiano[†3], Nikos Parotsidis[4], and Przemysław Uznański[5]

1 **University of Ioannina, Ioannina, Greece**
loukas@cs.uoi.gr
2 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
daniel.graf@inf.ethz.ch
3 **University of Rome Tor Vergata, Roma, Italy**
giuseppe.italiano@uniroma2.it
4 **University of Rome Tor Vergata, Roma, Italy**
nikos.parotsidis@uniroma2.it
5 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
przemyslaw.uznanski@inf.ethz.ch

Abstract

In the 2-reachability problem we are given a directed graph G and we wish to determine if there are two (edge or vertex) disjoint paths from u to v, for given pair of vertices u and v. In this paper, we present an algorithm that computes 2-reachability information for all pairs of vertices in $\mathcal{O}(n^\omega \log n)$ time, where n is the number of vertices and ω is the matrix multiplication exponent. Hence, we show that the running time of all-pairs 2-reachability is only within a log factor of transitive closure. Moreover, our algorithm produces a witness (i.e., a separating edge or a separating vertex) for all pair of vertices where 2-reachability does not hold. By processing these witnesses, we can compute all the edge- and vertex-dominator trees of G in $\mathcal{O}(n^2)$ additional time, which in turn enables us to answer various connectivity queries in $\mathcal{O}(1)$ time. For instance, we can test in constant time if there is a path from u to v avoiding an edge e, for any pair of query vertices u and v, and any query edge e, or if there is a path from u to v avoiding a vertex w, for any query vertices u, v, and w.

1998 ACM Subject Classification E.1 Graphs and Networks, F.2.2 Computations on Discrete Structures, G.2.2 Graph Algorithms

Keywords and phrases 2-reachability, All Dominator Trees, Directed Graphs, Boolean Matrix Multiplication

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.74

1 Introduction

The *all-pairs reachability problem* consists of preprocessing a directed graph (digraph) $G = (V, E)$ so that we can answer queries that ask if a vertex y is reachable from a vertex x. This problem has many applications, including databases, geographical information systems, social networks, and bioinformatics [11]. A classic solution to this problem is to compute the transitive closure matrix of G, either by performing a graph traversal (e.g., depth-first or breadth-first search) once per each vertex as source, or via matrix multiplication. For a

* A full version of the paper is available at https://arxiv.org/abs/1612.08075.
† Partially supported by MIUR, the Italian Ministry of Education, University and Research, under Project AMANDA (Algorithmics for MAssive and Networked DAta).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 74; pp. 74:1–74:14


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

digraph with n vertices and m edges, the former solution runs in $\mathcal{O}(mn)$ time, while the latter in $\mathcal{O}(n^\omega)$, where ω is the matrix multiplication exponent [4, 13, 17]. Here we study a natural generalization of the all-pairs reachability problem, that we refer to as *all-pairs 2-reachability*, where we wish to preprocess G so that we can answer fast the following type of queries: For a given vertex pair $x, y \in V$, are there two edge-disjoint (resp., internally vertex-disjoint) paths from x to y? Equivalently, by Menger's theorem [15], we ask if there is an edge $e \in E$ (resp., a vertex $z \in V$) such that there is no path from x to y in $G \setminus e$ (resp., $G \setminus z$). We call such an edge (resp., vertex) *separating* for the pair x, y.

One solution to the all-pairs 2-reachability problem is to compute all the dominator trees of G, with each vertex as source. The dominator tree of G with start vertex s is a tree rooted at s, such that a vertex v is an ancestor of a vertex w if and only if all paths from s to w include v [14]. All the separating edges and vertices for a pair s, v, appear on the path from s to v in the dominator tree rooted at s, in the same order as they appear in any path from s to v in G. Given all the dominator trees, we can process them to compute the 2-reachability information for all pairs of vertices (see Section 6). Since a dominator tree can be computed in $\mathcal{O}(m)$ time [2, 3], the overall running time of this algorithm is $\mathcal{O}(mn)$.

Our Results. In this paper, we show how to beat the $\mathcal{O}(nm)$ bound for dense graphs. Specifically, we present an algorithm that computes 2-reachability information for all pairs of vertices in $\mathcal{O}(n^\omega)$ time in a strongly connected digraph, and in $\mathcal{O}(n^\omega \log n)$ time in a general digraph. Hence, we show that the running time of all-pairs 2-reachability is only within a log factor of transitive closure. This result is tight up to a log factor, since it can be shown that all-pairs 2-reachability is at least as hard as computing the transitive closure, which is asymptotically equivalent to Boolean matrix multiplication [6]. Moreover, our algorithm produces a witness (separating edge or vertex) whenever 2-reachability does not hold. By processing these witnesses, we can find all the dominator trees of G in $\mathcal{O}(n^2)$ additional time. Thus, we also show how to compute all the dominator trees of a digraph in $\mathcal{O}(n^\omega \log n)$ time (in $\mathcal{O}(n^\omega)$ time if the graph is strongly connected), which improves the previously known $\mathcal{O}(mn)$ bound for dense graphs. This in turn enables us to answer various connectivity queries in $\mathcal{O}(1)$ time. E.g., we can test in $\mathcal{O}(1)$ time if there is a path from u to v avoiding an edge e, for any pair of query vertices u and v, and any query edge e, or if there is a path from u to v avoiding a vertex w, for any query vertices u, v, and w. We can also report all the edges or vertices that appear in all paths from u to v, for any query vertices u and v.

Related Work. To the best of our knowledge, ours is the first work that considers the all-pairs 2-reachability problem and gives a fast algorithm for it. In recent work Georgiadis et al. [9] investigate the effect of an edge or a vertex failure in a digraph G with respect to strong connectivity. Specifically, they show how to preprocess G in $\mathcal{O}(m + n)$ time in order to answer various sensitivity queries regarding strong connectivity in G under an arbitrary edge or vertex failure. For instance, they can compute in $\mathcal{O}(n)$ time the strongly connected components (SCCs) that remain in G after the deletion of an edge or a vertex, or report various statistics such as the number of SCCs in constant time per query (failed) edge or vertex. This result, however, cannot be applied for the solution of the 2-reachability problem. The reason is that if the deletion of an edge e leaves two vertices u and v in different SCCs in $G \setminus e$, the algorithm of [9] is not able to distinguish if there is still a path or no path from u to v in $G \setminus e$. Previously, King and Sagert [12] gave an algorithm that can quickly answer sensitivity queries for reachability in a directed acyclic graph (DAG) [12]. Specifically, they show how to process a DAG G so that, for any pair of query vertices x and y, and a query

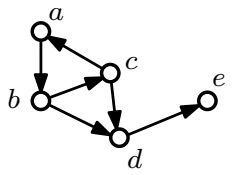
edge e, one can test in constant time if there is a path from x to y in $G \setminus e$. Note that the result of King and Sagert does not yield an efficient solution to the all-pairs 2-reachability problem, since we need $\mathcal{O}(m)$ queries just to find if there is a separating edge for a single pair of vertices. Moreover, their preprocessing time is $\mathcal{O}(n^3)$. Another interesting fact that arises from our work is that, somewhat surprisingly, computing all dominator trees in dense graphs is currently faster than computing a spanning arborescence from each vertex. The best algorithm for this problem is given by Alon et al. [1], who studied the problem of constructing a BFS tree from every vertex, and gave an algorithm that runs in $\mathcal{O}(n^{(3+\omega)/2})$ time.

Our Techniques. Our result is based on two novel approaches, one for DAGs and one for strongly connected digraphs. For DAGs we develop an algebra that operates on paths. We then use some version of 1-superimposed coding to apply our path algebra in a divide and conquer approach. This allows us to use Boolean matrix multiplication, in a similar vein to the computation of transitive closure. Unfortunately, our algebraic approach does not work for strongly connected digraphs. In this case, we exploit dominator trees in order to transform a strongly connected digraph G into two auxiliary graphs, so as to reduce 2-reachability queries in G to 1-reachability queries in those auxiliary graphs. This reduction works only for strongly connected digraphs and does not carry over to general digraphs. Our algorithm for general digraphs is obtained via a careful combination of those two approaches.

2 Preliminaries

We assume that the reader is familiar with standard graph terminology, as contained for instance in [5]. Let $G = (V, E)$ be a directed graph (digraph). Given an edge $e = (x, y)$ in E, we denote x (resp., y) as the *tail* (resp., *head*) *of* e. A *directed path* in G is a sequence of vertices v_1, v_2, ..., v_k, such that edge $(v_i, v_{i+1}) \in E$ for $i = 1, 2, \ldots, k-1$. The path is said to contain vertex v_i, for $i = 1, 2, \ldots, k$, and edge (v_i, v_{i+1}), for $i = 1, 2, \ldots, k-1$. The *length* of a directed path is given by its number of edges. As a special case, there is a path of length 0 from each vertex to itself. We write $u \rightsquigarrow v$ to denote that there is a path from u to v, and $u \not\rightsquigarrow v$ if there is no path from u to v. A *directed cycle* is a directed path, with length greater than 0, starting and ending at the same vertex. A *directed acyclic graph* (in short *DAG*) is a digraph with no cycles. A DAG has a *topological ordering*, i.e., a linear ordering of its vertices such that for every edge (u, v), u comes before v in the ordering (denoted by $u < v$). A digraph G is *strongly connected* if there is a directed path from each vertex to every other vertex. The *strongly connected components* of a digraph are its maximal strongly connected subgraphs. Given a subset of vertices $V' \subset V$, we denote by $G \setminus V'$ the digraph obtained after deleting all the vertices in V', together with their incident edges. Given a subset of edges $E' \subset E$, we denote by $G \setminus E'$ the digraph obtained after deleting all the edges in E'.

2-Reachability and 2-Reachability closure. We write $u \rightsquigarrow_{2e} v$ (resp., $u \rightsquigarrow_{2v} v$) to denote that there are two *edge-disjoint* (resp., internally *vertex-disjoint*) paths from u to v, and $u \not\rightsquigarrow_{2e} v$ (resp., $u \not\rightsquigarrow_{2v} v$) otherwise. As a special case, we assume that $v \rightsquigarrow_{2e} v$ (resp., $v \rightsquigarrow_{2v} v$) for each vertex v in G. We define an abstract set $E^+ = E \cup \{\top, \bot\}$. The semantic of this set is as follows: $e \in E$ corresponds to an edge e separating two vertices, $\top$ corresponds to $\rightsquigarrow_{2e}$ (there is no single separating edge) and $\bot$ corresponds to $\not\rightsquigarrow$ (there is no path). Given a digraph G,



$$\begin{bmatrix} \top & (a,b) & (a,b) & (a,b) & (a,b) \\ (b,c) & \top & (b,c) & \top & (d,e) \\ (c,a) & (c,a) & \top & \top & (d,e) \\ \bot & \bot & \bot & \top & (d,e) \\ \bot & \bot & \bot & \bot & \top \end{bmatrix}$$

Figure 1 A graph and its (not unique) 2-reachability closure matrix.

we define a 2-*reachability closure* of G, denoted by $G^{\rightsquigarrow 2e}$, to be a matrix such that:

$$G^{\rightsquigarrow 2e}[u,v] \stackrel{\text{def}}{=} \begin{cases} \top & \text{if } u \rightsquigarrow_{2e} v \\ \bot & \text{if } u \not\rightsquigarrow v \\ e & \text{where } e \text{ is any separating edge for } u \text{ and } v. \end{cases}$$

Since $v \rightsquigarrow_{2e} v$ for each $v \in V$, $G^{\rightsquigarrow 2e}[v,v] = \top$. An example of a graph with a 2-reachability closure matrix is given in Figure 1. Note that a 2-reachability closure matrix is not necessarily unique, as there might be multiple separating edges for a given vertex pair. We define the 2-*reachability left closure* $G_L^{\rightsquigarrow 2e}$ by replacing *any separating edge* with *first separating edge* and the 2-*reachability right closure* $G_R^{\rightsquigarrow 2e}$ by replacing it with *last separating edge*.

Note that if there is only one edge separating u and v, then $G^{\rightsquigarrow 2e}[u,v] = G_L^{\rightsquigarrow 2e}[u,v] = G_R^{\rightsquigarrow 2e}[u,v]$. Given any 2-reachability closure matrix, one can compute efficiently the 2-reachability left and right closure matrices. We sketch below the basic idea for the left closure (the right closure is completely symmetric). Let u and v be any two vertices. If $G^{\rightsquigarrow 2e}[u,v]$ is either $\top$ or $\bot$, then $G_L^{\rightsquigarrow 2e}[u,v] = G^{\rightsquigarrow 2e}[u,v]$. Otherwise, let $G^{\rightsquigarrow 2e}[u,v] = (x,y)$: if $u \rightsquigarrow_{2e} x$ (i.e., if $G^{\rightsquigarrow 2e}[u,x] = \top$) then (x,y) is the first separating edge for u and v and $G_L^{\rightsquigarrow 2e}[u,v] = (x,y)$; otherwise, $u \not\rightsquigarrow_{2e} x$ (i.e., $G^{\rightsquigarrow 2e}[u,x] \neq \top$) and $G_L^{\rightsquigarrow 2e}[u,v] = G_L^{\rightsquigarrow 2e}[u,x]$. We show how to compute $G_L^{\rightsquigarrow 2e}$ and $G_R^{\rightsquigarrow 2e}$ from $G^{\rightsquigarrow 2e}$ in a total of $\mathcal{O}(n^2)$ worst-case time.

3 All-pairs 2-reachability in DAGs

In this section we present our $\mathcal{O}(n^\omega \log n)$ time algorithm for all-pairs 2-reachability in DAGs. The high-level idea is to mimic the way Boolean matrix multiplication can be used to compute the transitive closure of a graph: recursively along a topological order, combine the transitive closure of the first and the second half of the vertices in a single matrix multiplication. However, while in transitive closure for each pair (i,j) we have to store only information on whether there is a path from i to j, for all-pairs 2-reachability this is not enough. First, we describe a path algebra, used by our algorithm to operate on paths between pairs of vertices in a concise manner. We then continue with the description of a matrix product-like operation, which is the backbone of our recursive algorithm. Finally, we show how to implement those operations efficiently using some binary encoding and decoding at every step of the recursion.

Before introducing our new algorithm, we need some terminology. Let $G = (V,E)$ be a DAG, and let E_1, E_2 be a partition of its edge set E, $E = E_1 \cup E_2$. We say that a partition is an *edge split* if there is no triplet of vertices x, y, z in G such that $(x,y) \in E_2$ and $(y,z) \in E_1$ simultaneously. Informally speaking, under such split, any path in G from a vertex u to a vertex v consists of a sequence of edges from E_1 followed by a sequence of edges from E_2 (as a special case, any of those sequences can be empty). We denote the edge split by $G = (V, E_1, E_2)$ (See Figure 2). We say that vertex x in $G = (V, E_1, E_2)$ is on the *left* (resp., *right*) *side* of the partition if x is adjacent only to edges in E_1 (resp., E_2). We assume without loss of generality that the vertices of G are given in a topological ordering $v_1, v_2, \ldots, v_n$.

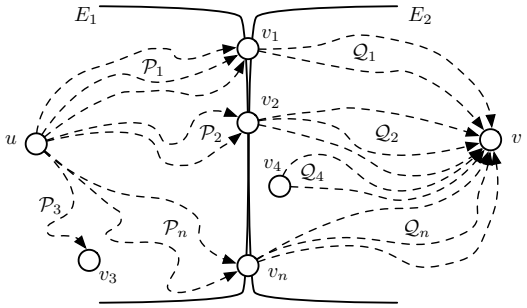


Figure 2 An edge split of a DAG $G = (V, E_1, E_2)$.

3.1 Algebraic approach

Consider a family of paths $\mathcal{P} = \{P_1, P_2, \ldots, P_\ell\}$, all sharing the same starting and ending vertices u and v. We would like to distinguish between the following three possibilities: (i) $\mathcal{P}$ is empty; (ii) at least one edge e belongs to every path $P_i \in \mathcal{P}$; or (iii) there is no edge that belongs to all paths in (nonempty) $\mathcal{P}$. To do that, we define the *representation* $\text{repr}(\mathcal{P})$:

$$\text{repr}(\mathcal{P}) \stackrel{\text{def}}{=} \bigcap_{i=1}^{\ell} P_i = \begin{cases} \mathbb{U} & \text{if } \mathcal{P} = \emptyset \\ \emptyset & \text{if no edge belongs to all } P_i \\ \{e \in E : e \in P_i,\ 1 \le i \le \ell\} & \text{otherwise.} \end{cases}$$

where $\mathbb{U}$ denotes the top symbol in the Boolean algebra of sets (i.e., the complement of $\emptyset$). We also define a *left representation* $\text{repr}_L(\mathcal{P}) \in E^+$, where $E^+ = E \cup \{\top, \bot\}$, as follows:

$$\text{repr}_L(\mathcal{P}) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } \mathcal{P} = \emptyset \\ \top & \text{if no edge belongs to all } P_i \\ e & \text{such that } e \in P_i, 1 \le i \le \ell, \text{ and } \text{tail}(e) \text{ is } \textit{minimum} \\ & \text{in the topological order} \end{cases}$$

A *right representation* $\text{repr}_R(\mathcal{P}) \in E^+$ is defined symmetrically to $\text{repr}_L(\mathcal{P})$, by replacing *minimum* with *maximum*. If $\text{repr}_L(\mathcal{P}) \in E$ (resp., $\text{repr}_R(\mathcal{P}) \in E$), we say that $\text{repr}_L(\mathcal{P})$ (resp., $\text{repr}_R(\mathcal{P})$) is the *first* (resp., *last*) *common edge* in $\mathcal{P}$. Note that if $\mathcal{P}$ is the set of *all the paths* from u to v, then $\text{repr}(\mathcal{P})$ contains all the information about $G^{\rightsquigarrow 2e}[u, v]$. Additionally, $G_L^{\rightsquigarrow 2e}[u, v] = \text{repr}_L(\mathcal{P})$ and $G_R^{\rightsquigarrow 2e}[u, v] = \text{repr}_R(\mathcal{P})$. With a slight abuse of notation we also say that $G^{\rightsquigarrow 2e}[u, v] \in \text{repr}(\mathcal{P})$.

▶ **Observation 1.** *Let $G = (V, E_1, E_2)$ be an edge split of a DAG, and let u and v be two arbitrary vertices in G. For $1 \le i \le n$, let $\mathcal{P}_i = \{P \subseteq E_1 : P$ is a path from u to $v_i\}$, and $\mathcal{Q}_i = \{Q \subseteq E_2 : Q$ is a path from v_i to $v\}$ (See Figure 2) and let $\mathcal{S}$ be the family of all paths from u to v. Then:* $\text{repr}(\mathcal{S}) = \bigcap_{i=1}^{n} \Big(\text{repr}(\mathcal{P}_i) \cup \text{repr}(\mathcal{Q}_i)\Big)$

A straightforward application of Observation (1) yields immediately a polynomial time algorithm for computing $G^{\rightsquigarrow 2e}$. However, this algorithm is not very efficient, since the size of $\text{repr}(\mathcal{P})$ can be as large as $(n-1)$. In the following we will show how to obtain a faster algorithm, by replacing $\text{repr}(\mathcal{P})$ with a suitable combination of $\text{repr}_L(\mathcal{P})$ and $\text{repr}_R(\mathcal{P})$.

We next define two operations, denoted as *serial* and *parallel*. Although those operations are formally defined on $E^+ = E \cup \{\top, \bot\}$, they have a more intuitive interpretation as

operations on path families. We start with the serial operation $\otimes$. For $a, b \in E^+$, we define:

$$a \otimes b \stackrel{\text{def}}{=} \begin{cases} (\bot, \bot) & \text{if } a = \bot \text{ or } b = \bot \\ (a, b) & \text{otherwise.} \end{cases}$$

We define $\oplus$ as the parallel operator. Namely, for arbitrary $a \in E^+$: $a \oplus \bot \stackrel{\text{def}}{=} a$, $\bot \oplus a \stackrel{\text{def}}{=} a$, $a \oplus \top \stackrel{\text{def}}{=} \top$, $\top \oplus a \stackrel{\text{def}}{=} \top$, and otherwise, for $e, e' \in E$:

$$e \oplus e' \stackrel{\text{def}}{=} \begin{cases} \top & \text{if } e \neq e' \\ e & \text{if } e = e' \end{cases}$$

We extend the definition of $\oplus$ to operate on elements of $E^+ \times E^+$, as follows: $(a_1, b_1) \oplus (a_2, b_2) \stackrel{\text{def}}{=} (a_1 \oplus a_2, b_1 \oplus b_2)$. Ideally, we want the operator $\oplus$ either to preserve consistently the first common edge or to preserve consistently the last common edge, under the union of path families. If for instance we preserve the first common edge, that means that if $\mathcal{P}$ and $\mathcal{P}'$ are two path families sharing the same endpoints then we want $\text{repr}_L(\mathcal{P} \cup \mathcal{P}') = \text{repr}_L(\mathcal{P}) \oplus \text{repr}_L(\mathcal{P}')$ to hold. However, this is not necessarily the case, as for example both $\mathcal{P}$ and $\mathcal{P}'$ could consist of a single path, with both paths sharing an intermediate edge e', but both with two different initial edges, respectively e_1 and e_2. Thus $\text{repr}_L(\mathcal{P}) \oplus \text{repr}_L(\mathcal{P}') = e_1 \oplus e_2 = \top$ while $\text{repr}_L(\mathcal{P} \cup \mathcal{P}') = e'$. As shown in the following lemma, this is not an issue if the path families considered are exhaustive in taking every possible path between a pair of vertices.

▶ **Lemma 2.** *Let $G, \mathcal{P}_i, \mathcal{Q}_i$ and $\mathcal{S}$ be as in Observation 1. Then:*
(a) $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (\bot, \bot)$ *iff* $\text{repr}(\mathcal{S}) = \mathbb{U}$*;*
(b) *if* $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (e_1, \top)$ *then* $\text{repr}(\mathcal{S}) \ni e_1$*;*
(c) *if* $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (\top, e_2)$ *then* $\text{repr}(\mathcal{S}) \ni e_2$*;*
(d) *if* $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (e_1, e_2)$ *then* $\text{repr}(\mathcal{S}) \ni e_1, e_2$*;*
(e) $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (\top, \top)$ *iff* $\text{repr}(\mathcal{S}) = \emptyset$*.*

We now consider the special case where one side of the partition defined in Observation 1 contains only paths of length one. In particular, we say that the edge set $E' \subseteq E$ is *thin*, if there exists no triplet of vertices x, y, z such that $(x, y) \in E'$ and $(y, z) \in E'$.

▶ **Lemma 3.** *Let $G, \mathcal{P}_i, \mathcal{Q}_i$ and $\mathcal{S}$ be as in Observation 1. Additionally, let E_1 be thin. Then*
(a) $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (\bot, \bot)$ *iff* $\text{repr}_R(\mathcal{S}) = \bot$*;*
(b) *if* $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (e_1, \top)$ *then* $\text{repr}_R(\mathcal{S}) = e_1$*;*
(c) *if* $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (\top, e_2)$ *then* $\text{repr}_R(\mathcal{S}) = e_2$*;*
(d) *if* $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (e_1, e_2)$ *then* $\text{repr}_R(\mathcal{S}) = e_2$*;*
(e) $\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i)) = (\top, \top)$ *iff* $\text{repr}_R(\mathcal{S}) = \top$*.*

We define the following projection operator π: $\pi(\bot, \bot) \stackrel{\text{def}}{=} \bot$, $\pi(\top, \top) \stackrel{\text{def}}{=} \top$, $\pi(e', e) = \pi(\top, e) = \pi(e, \top) \stackrel{\text{def}}{=} e$. With this new terminology, Lemma 2 and Lemma 3 can be simply restated as follows:

▶ **Corollary 4.** *Let $G, \mathcal{P}_i, \mathcal{Q}_i$ and $\mathcal{S}$ be as in Observation 1. Then*
(i) $\pi(\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i))) = \top$ *iff* $\text{repr}(\mathcal{S}) = \emptyset$*,*
(ii) $\pi(\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i))) = \bot$ *iff* $\text{repr}(\mathcal{S}) = \mathbb{U}$*, and*
(iii) $\pi(\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i))) \in \text{repr}(\mathcal{S})$ *otherwise.*

▶ **Corollary 5.** *Let $G, \mathcal{P}_i, \mathcal{Q}_i$ and $\mathcal{S}$ be as in Observation 1, and let E_1 be thin. Then* $\pi(\bigoplus_{i=1}^{n}(\text{repr}_L(\mathcal{P}_i) \otimes \text{repr}_R(\mathcal{Q}_i))) = \text{repr}_R(\mathcal{S})$*.*

Matrix product. Now we define a *path-based matrix product* based on the previously defined operators: $(A \circ B)[i,j] \stackrel{\text{def}}{=} \pi\big(\bigoplus_k A[i,k] \otimes B[k,j]\big)$. Throughout, we assume that the vertices of G are sorted according to a topological ordering. In the following lemma $\mathbf{B}$ represents a thin set of edges (i.e., the set of edges from a subset of vertices to another disjoint subset of vertices).

▶ **Lemma 6.** *Let* $\begin{bmatrix} A & B \\ 0 & C \end{bmatrix}$ *be the adjacency matrix of a DAG* $G = (V, E)$*, where* A, B *and* C *are respectively* $k \times k$*,* $k \times (n-k)$ *and* $(n-k) \times (n-k)$ *submatrices. If* $\mathbf{B}$ *is the matrix containing* $\perp$ *for every* 0 *in* B *and the appropriate* $e \in E$ *for every* 1 *in* B*, then:*

$$\begin{bmatrix} A_L^{\rightsquigarrow 2e} & A_L^{\rightsquigarrow 2e} \circ (\mathbf{B} \circ C_R^{\rightsquigarrow 2e}) \\ \perp & C_R^{\rightsquigarrow 2e} \end{bmatrix}$$

is a 2-reachability closure of G *(not necessarily unique).*

By Lemma 6, the 2-reachability closure can be computed by performing path-based matrix products on the left and right 2-reachability closures of smaller matrices. This gives immediately a recursive algorithm for computing the 2-reachability closure: indeed, as already shown in Section 2, one can compute the left and right 2-reachability closures in $\mathcal{O}(n^2)$ time from any 2-reachability closure. In the next section we show how to implement this recursion efficiently by describing how to compute efficiently path-based matrix products.

3.2 Encoding and decoding for Boolean matrix product

We start this section by showing how to efficiently compute path-based matrix products using Boolean matrix multiplications. The first step is to encode each entry of the matrix as a bitword of length $8k$ where $k = \lceil \log_2(n+1) \rceil$. We use Boolean matrix multiplication of matrices of bitwords, with bitwise AND/OR operations, denoted respectively with symbols $\wedge$ and $\vee$. Our bitword length is $\mathcal{O}(\log n)$, so matrix multiplication takes $\mathcal{O}(n^\omega \log n)$ time by performing Boolean matrix multiplication for each coordinate separately.

We make use of the fact that after each multiplication we can afford a post-processing phase, where we perform actions which guarantee that the resulting bitwords represent a valid 2-reachability closure.

First, we note that when encoding a specific matrix, we know whether it is used as a left-side or a right-side component of multiplication. The main idea is to encode left-side and right-side $\perp$ as $\{0\}^{8k}$, left-side and right-side $\top$ as $\{1\}^{8k}$. For any other value, append $\{1\}^{4k}$ as a prefix or suffix (depending on whether it is used as a left-side or right-side component), to the encoding of an edge. The encoding of an edge is a simple 1-superimposed code: concatenation of the edge ID and complement of the edge ID. To be more precise, whenever a bitword represents an edge e in a left-closure, then it is of the form $\mathrm{ID}_e\overline{\mathrm{ID}_e}\{1\}^{4k}$; whenever a bitword represents an edge e in a right-closure, then it is of the form $\{1\}^{4k}\mathrm{ID}_e\overline{\mathrm{ID}_e}$, where $\overline{w}$ denotes the complement of bitword w.

The serial operator $\otimes$ is implemented by coordinate-wise AND over two bitwords. Recall that the operator $\otimes$ always has as its first (left) operand an element from a left-closure matrix and as its second (right) operand an element from a right-closure. It is easy to verify that result of AND is a concatenation of two bitwords of length $4k$ encoding either $\perp, \top$ or $e \in E$. We observe that $\otimes$ is calculated properly in all cases: (let $e, e_1, e_2 \in E, e_1 \neq e_2$)

1. $e \otimes \top = (e, \top)$ since $\mathrm{ID}_e\overline{\mathrm{ID}_e}\{1\}^{4k} \wedge \{1\}^{8k} = \mathrm{ID}_e\overline{\mathrm{ID}_e}\{1\}^{4k}$,
2. $\top \otimes e = (e, \top)$ since $\{1\}^{8k} \wedge \{1\}^{4k}\mathrm{ID}_e\overline{\mathrm{ID}_e} = \{1\}^{4k}\mathrm{ID}_e\overline{\mathrm{ID}_e}$,
3. $e_1 \otimes e_2 = (e_1, e_2)$ since $\mathrm{ID}_{e_1}\overline{\mathrm{ID}_{e_1}}\{1\}^{4k} \wedge \{1\}^{4k}\mathrm{ID}_{e_2}\overline{\mathrm{ID}_{e_2}} = \mathrm{ID}_{e_1}\overline{\mathrm{ID}_{e_1}}\mathrm{ID}_{e_2}\overline{\mathrm{ID}_{e_2}}$,
4. $e \otimes \perp = \top \otimes \perp = \perp \otimes \perp = \perp \otimes e = \perp \otimes \top = (\perp, \perp)$ since $\{0,1\}^{8k} \wedge \{0\}^{8k} = \{0\}^{8k}$,
5. $\top \otimes \top = (\top, \top)$ since $\{1\}^{8k} \wedge \{1\}^{8k} = \{1\}^{8k}$.

The parallel operator $\oplus$ is implemented as coordinate-wise OR over bitwords of length $8k$. Note that all bitwords can be binary representations of pairs of elements in E^+ of the form $(e_1, e_2), (e_1, \top), (\top, e_2), (\bot, \bot), (\top, \top)$, since only those forms appear as a result of an $\otimes$ operation. Recall that $\oplus$ satisfies $(a_1, b_1) \oplus (a_2, b_2) = (a_1 \oplus a_2, b_1 \oplus b_2)$, thus w.l.o.g. it is enough to verify the correctness of the implementation over the first $4k$ bits of encoding. Observe that all cases, except when both bitwords include encoded edges, are managed correctly by the execution of coordinate-wise OR: (let $e \in E$)

1. $\bot \oplus \bot = \bot$ since $\{0\}^{4k} \vee \{0\}^{4k} = \{0\}^{4k}$,
2. $\bot \oplus e = e \oplus \bot = e$ since $\mathrm{ID}_e\overline{\mathrm{ID}_e} \vee \{0\}^{4k} = \mathrm{ID}_e\overline{\mathrm{ID}_e}$,
3. $\bot \oplus \top = \top \oplus \bot = \top$ since $\{1\}^{4k} \vee \{0\}^{4k} = \{1\}^{4k}$,
4. $e \oplus \top = \top \oplus e = \top$ since $\mathrm{ID}_e\overline{\mathrm{ID}_e} \vee \{1\}^{4k} = \{1\}^{4k}$.

We are only left to take care of operations of the form $e_1 \oplus e_2$ for $e_1, e_2 \in E$. According to the definition of the parallel operator $\oplus$, we would like $e_1 \oplus e_2 = e \in E$ iff $e_1 = e_2 = e$ and otherwise $e_1 \oplus e_2 = \top$. This special case is handled by the fact that we encode edges using 1-superimposed codes. That is, the binary representation of ID_e has the property that $\mathrm{ID}_e[1 \,..\, 2k] = \overline{\mathrm{ID}_e[2k+1 \,..\, 4k]}$. Moreover, the coordinate-wise OR of two encodings of edges, that is $\mathrm{X} = \mathrm{ID}_{e_1} \vee \mathrm{ID}_{e_2}$, has such property iff $e_1 = e_2$. Thus in order to successfully decode the result of chained $\oplus$ from coordinate-wise OR, we need to distinguish the following cases (our result is encoded as $\mathrm{X} = \mathrm{X}[1 \,..\, 2k]\mathrm{X}[2k+1 \,..\, 4k]$):

1. $\mathrm{X} = \{0\}^{4k}$, then the result is $\bot$,
2. $\mathrm{X}[1 \,..\, 2k] = \overline{\mathrm{X}[2k+1 \,..\, 4k]}$, then X is the encoding of the resulting edge,
3. otherwise the result is $\top$.

The implementation of a projection operator follows trivially.

The operations needed to compute the l-th coordinate of all entries of the final path-based matrix product (before decoding of entries) can be implemented as a Boolean matrix product of the l-th coordinate of the entries of $A_L^{\rightsquigarrow 2e}$ and $B_R^{\rightsquigarrow 2e}$. All the tools developed in this section allow us to compute the 2-reachability closure for DAGs. Our recursive algorithm follows closely Lemma 6.

▶ **Lemma 7.** *Given a DAG with n vertices, its 2-reachability closure can be computed in time $\mathcal{O}(n^\omega \log n)$.*

4 All-pairs 2-reachability in strongly connected graphs

In this section we focus on strongly connected graphs. In this case reachability is simple: for any pair of vertices $(u, v) \in V \times V$ we have $u \rightsquigarrow v$ in G. But in case that $u \not\rightsquigarrow_{2e} v$ in G, finding a separating edge that appears in all paths from u to v in G can still be a challenge. We show that we can report such an edge in constant time after $\mathcal{O}(n^\omega)$ preprocessing. The main result of this section is the following theorem.

▶ **Theorem 8.** *The 2-reachability closure of a strongly connected graph can be computed in time $\mathcal{O}(n^\omega)$.*

Our construction is based on the notion of auxiliary graph and it will be given in Section 4.3. Its running time will be analyzed in Lemma 13 and its correctness hinges on Lemma 15.

4.1 Reduction to two single-source problems

Let $G = (V, E)$ be a strongly connected digraph. Let s be a fixed but arbitrary vertex of G. The proof of the following lemma is immediate.

▶ **Lemma 9.** *For any pair of vertices u and v: If there is an edge $e \in E(G)$ such that $u \not\to v$ in $G \setminus e$, then either $u \not\to s$ in $G \setminus e$ or $s \not\to v$ in $G \setminus e$.*

Let $\mathcal{P}_{u,s}$ be the family of all paths from u to s and let $\mathcal{P}_{s,v}$ be the family of all paths from s to v. We denote by e_u the first edge on all paths in $\mathcal{P}_{u,s}$, and by e_v the last edge on all paths in $\mathcal{P}_{s,v}$. Note that there might be no edge that is on all paths of $\mathcal{P}_{u,s}$: in this case we say that e_u does not exist. If there are several edges on all paths in $\mathcal{P}_{u,s}$, then they are totally ordered, so it is clear which is the *first* edge (similarly for e_v and $\mathcal{P}_{s,v}$). We now show that in order to search for a separation witness for (u, v), it suffices to focus on e_u and e_v.

▶ **Lemma 10.** *If there is some e such that $u \not\to v$ in $G \setminus e$, then at least one of the following statements is true:*

- *e_u exists and $u \not\to v$ in $G \setminus e_u$.*
- *e_v exists and $u \not\to v$ in $G \setminus e_v$.*

Hence, in order to check whether there is an edge that separates u from v in G, it suffices to look at the reachability information in $G \setminus e_u$ (a graph which does not depend on u) and at the reachability information in $G \setminus e_v$ (a graph which does not depend on v). Unfortunately, this is not enough to derive an efficient algorithm, since we would have still to look at as many as $2n$ different graphs (as we explain later, and as it was first shown in [10], there can be at most $2n - 2$ edges whose removal can affect the strong connectivity of the graph). As a result, computing the transitive closures of all those graphs would require $\mathcal{O}(n^{\omega+1})$ time. The key insight to reduce the running time to $\mathcal{O}(n^{\omega})$ is to construct an auxiliary graph H, whose reachability is identical to $G \setminus e_v$ for any query pair (u, v), and a second auxiliary graph H' whose reachability is identical to $G \setminus e_u$ for any query pair (u, v). Note that the edge that is missing from the graph depends always on one of the two endpoints of the reachability query. As a consequence, we have to consider only n^2 and not n^3 different queries for H and H'.

4.2 Strong bridges and dominator tree decomposition

Before we construct these auxiliary graphs, we need some more terminology and prior results.

Flow graphs, dominators, and bridges. A *flow graph* $G_s = (V, E, s)$ is a digraph with a distinguished *start vertex* s. We denote by $G_s^R = (V, E^R, s)$ the reverse flow graph of G_s; the graph resulted by reversing the direction of all edges $e \in E$. Vertex u is a *dominator* of a vertex v (u *dominates* v) if every path from s to v in G_s contains u; u is a *proper dominator* of v if u dominates v and $u \neq v$. The dominator relation is reflexive and transitive. Its transitive reduction is a rooted tree, the *dominator tree* D: u dominates v if and only if u is an ancestor of v in D. If $v \neq s$, the parent of v in D, denoted by $d(v)$, is the *immediate dominator* of v: it is the unique proper dominator of v that is dominated by all proper dominators of v. For any vertex v, we let $D(v)$ denote the set of descendants of v in D, i.e., the vertices dominated by v. Dominators can be computed in linear time [2, 3, 7]. An edge (x, y) is a *bridge* of the flow graph G_s if all paths from s to y include (x, y).

Strong bridges. Let $G = (V, E)$ be a strongly connected digraph. An edge e of G is a *strong bridge* if $G \setminus e$ is no longer strongly connected. Let s be an arbitrary start vertex of G. Since G is strongly connected, all vertices are reachable from s and reach s, so we can view both G and G^R as flow graphs with start vertex s, denoted respectively by G_s and G^R_s.

▶ **Property 11** ([10]). *Let s be an arbitrary start vertex of G. An edge $e = (x, y)$ is a strong bridge of G if and only if it is a bridge of G_s or a bridge of G^R_s (or both).*

As a consequence of Property 11, all the strong bridges of the digraph G can be obtained from the bridges of the flow graphs G_s and G^R_s, and thus there can be at most $(2n-2)$ strong bridges in a digraph G. Using the linear time algorithms for computing dominators, we can thus compute all strong bridges of G in time $\mathcal{O}(m + n) \subseteq \mathcal{O}(n^\omega)$. We use the following lemma from [8] that holds for a flow graph G_s of a strongly connected digraph G.

▶ **Lemma 12** ([8]). *Let G be a strongly connected digraph and let (x, y) be a strong bridge of G. Also, let D and D^R be the dominator trees of the corresponding flow graphs G_s and G^R_s, respectively, for an arbitrary start vertex s.*

(a) *Suppose $x = d(y)$. Let w be any vertex that is not a descendant of y in D. Then there is a path from w to x in G avoiding all proper descendants of y in D. Moreover, all paths in G from w to any descendant of y in D contain the edge $(d(y), y)$.*

(b) *Suppose $y = d^R(x)$. Let w be any vertex that is not a descendant of x in D^R. Then there is a path from x to w in G avoiding all proper descendants of x in D^R. Moreover, all paths in G from any descendant of x in D^R to w contain the edge $(x, d^R(x))$.*

Bridge decomposition. After deleting from the dominator trees D and D^R respectively the bridges of G_s and G^R_s, we obtain the *bridge decomposition* of D and D^R into forests $\mathcal{D}$ and $\mathcal{D}^R$. Throughout this section, we denote by T_v (resp., T^R_v) the tree in $\mathcal{D}$ (resp., $\mathcal{D}^R$) containing vertex v, and by r_v (resp., r^R_v) the root of T_v (resp., T^R_v). Given a digraph $G = (V, E)$, and a set of vertices $S \subseteq V$, we denote by $G[S]$ the subgraph induced by S. In particular, $G[D(r)]$ denotes the subgraph induced by the descendants of vertex r in D.

4.3 Overview of the algorithm and construction of auxiliary graphs

The high-level idea of our algorithm is to compute two auxiliary graphs H and H' from G and G^R, respectively, with the following property: Given two vertices u and v, we have that $u \rightsquigarrow_{2e} v$ in G if and only $u \rightsquigarrow v$ in H and $v \rightsquigarrow u$ in H'. To construct the auxiliary graphs H and H', we use the bridge decompositions of D and D^R, respectively.

The two extremal edges e_u and e_v, defined in Section 4.1, can be also defined in terms of the bridge decompositions. In particular, e_v is the bridge entering the tree T_v of the bridge decomposition of D, so $e_v = (d(r_v), r_v)$, and e_u is the reverse bridge entering the tree D^R_u of the bridge decomposition of D^R, so $e_u = (r^R_u, d^R(r^R_u))$. Hence if there exists a path from u to v avoiding each of the strong bridges e_v and e_u, then $u \rightsquigarrow_{2e} v$ in G. By Lemma 10, it is enough if H models the reachability of $G \setminus e_v$ and H' the reachability of $G \setminus e_u$. So H is responsible for answering whether u has a path to v avoiding e_v, while H' is responsible for answering whether u has a path to v avoiding e_u. Then, if any of the reachability queries in H and H' returns false, we immediately have an edge that appears in all paths from u to v.

We next show to compute the auxiliary graphs H and H' in $\mathcal{O}(n^2)$ time. In particular, the auxiliary graph $H = (V, E')$ of the flow graph $G_s = (V, E, s)$ is constructed as follows. Initially, $E' = E \setminus BR$, where BR is the set of bridges of G_s. For all bridges (p, q) of G_s do the following: For each edge $(x, y) \in E$ such that $x \in D(q), y \notin D(q)$, we add the edge (p, y)

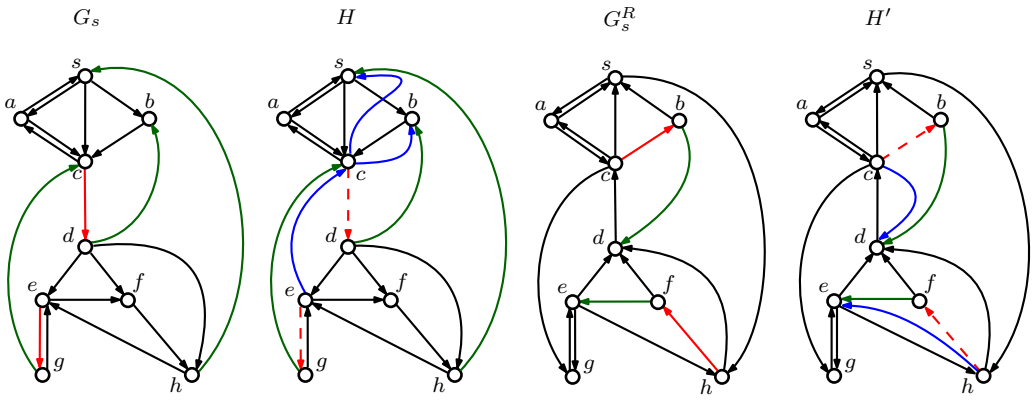


Figure 3 Auxiliary graphs H and H' which are derived from G_s and G_s^R, respectively. The deleted edges, the bridges of G_s and G_s^R, are shown in red, the newly added edges are shown in blue. The blue edges are drawn along the green edges from G_s and G_s^R which are the reason for their insertion. Here we see, that for example b is 2-reachable from e, since there are two (edge and vertex) disjoint paths (e, g, c, d, b) and (e, f, h, s, b) in G. In H, e reaches b through the path (e, c, b), and in H', b reaches e through the path (b, s, h, e). We also see, that edge (c, d) separates a and f in G, and even though f reaches a in H' through the path (f, d, c, a), a does not reach f in H. To illustrate why both H and H' are relevant in Lemma 15, consider the following example: vertex c is unreachable from b in $G \setminus (b, c)$, which we also detect as there is no c-b path in H' (even though there is a b-c path in H).

in E', i.e., we set $E' = E' \cup (p, y)$. A detailed implementation can be found in full version of the paper. Together with graph H, the algorithm outputs an array of edges ("witnesses") W, such that for each vertex $v \neq s$, $W[v] = (d(r_v), r_v)$ is a candidate separating edge for v and any other vertex. The computation of H' is completely analogous.

Once H and H' are computed, their transitive closure can be computed in $\mathcal{O}(n^\omega)$ time, after which reachability queries can be answered in constant time. Thus, we can preprocess a strongly connected digraph G in total time $\mathcal{O}(n^\omega)$ and answer 2-reachability queries in constant time, as claimed by Theorem 8.

▶ **Lemma 13.** *The auxiliary graph H can be computed in $\mathcal{O}(n^2)$ time and space.*

▶ **Lemma 14.** *For all $w \in V$, no edge $(x, y) \in E(H)$ exists with $x \notin D(r_w)$ and $y \in D(r_w)$.*

To show the correctness of our approach, we consider queries where we are given an ordered pair of vertices (u, v), and we wish to return whether there exists an edge e such that $u \not\rightsquigarrow v$ in $G \setminus e$. We can answer this query in constant time by answering the queries $u \rightsquigarrow v$ in H and $v \rightsquigarrow u$ in H'. Given Lemma 10, it is sufficient to prove the following:

▶ **Lemma 15.** *The auxiliary graphs H and H' satisfy these two conditions:*
- *If e_v exists, then $u \rightsquigarrow v$ in $G \setminus e_v$ if and only if $u \rightsquigarrow v$ in H.*
- *If e_u exists, then $u \rightsquigarrow v$ in $G \setminus e_u$ if and only if $v \rightsquigarrow u$ in H'.*

5 All-pairs 2-reachability in general graphs

In this section, we show how to compute the 2-reachability of a general digraph by suitably combining the previous algorithms for DAGs and for strongly connected digraphs. First, note that the 2-reachability closure of a strongly connected graph G can be constructed as follows: $G^{\rightsquigarrow 2e}[i, j] = \top$ if i has two edge-disjoint paths to j and $G^{\rightsquigarrow 2e}[i, j] \in E$ if there is

an edge $e \in E$ such that $i \not\rightsquigarrow j$ in $G \setminus e$. No entry of $G^{\rightsquigarrow 2e}$ contains $\perp$ since G is strongly connected. After $\mathcal{O}(n^\omega)$ time preprocessing all the above queries can be answered in constant time. Therefore, the 2-reachability closure can be computed in $\mathcal{O}(n^\omega)$ time.

Let G be a general digraph. The condensation of G is the DAG resulting after the contraction of every strongly connected component of G into a single vertex. We assume, without loss of generality, that the vertices are ordered as follows: The vertices in the same strongly connected component of G appear consecutively in an arbitrary order, and the strongly connected components are ordered with respect to the topological ordering of the condensation of G. Moreover, we assume that we have access to a function `stronglyConnected`(u, v) that answers whether the vertices u and v are strongly connected.

The key insight is that every idea presented in Section 3 never truly used the fact that the input graph is a DAG, just the properties of an edge split, that is finding edge partition into two sets so that no vertex has incoming edge from second set and outgoing edge from first set simultaneously. If we are able to extend the definition of an edge split to a general graph in a way highlighted above, and the definitions of $\text{repr}(), \text{repr}_R()$ and $\text{repr}_L()$, then all of the results from Section 3 carry over to a general graph G. Note that given *arbitrary* path family $\mathcal{P}$, $\text{repr}_L(\mathcal{P})$ and $\text{repr}_R(\mathcal{P})$ might be ill-defined, since paths in an arbitrary path family might not share the order of common edges. However, we are only using this notation for path families containing exactly all of paths connecting a given pair of vertices in the graph: for such families, the order of common edges is shared.

The high-level idea behind our approach is to extend the 2-reachability closure algorithm for DAGs, as follows. At each recursive call, the algorithm attempts to find a balanced separation of the set of vertices, with respect to their fixed precomputed order, into two sets such that there is no pair across the two sets that is strongly connected. If such a balanced separation can be found, then the instance is (roughly) equally divided into two instances. Otherwise, if there is no balanced separation of the set of vertices into two subsets, then one of the following properties holds: (i) the larger instance is a strongly connected component, or (ii) the recursive call on the larger instance separates a large strongly connected component, on which we can compute the 2-reachability closure in $\mathcal{O}(n^\omega)$ time.

▶ **Theorem 16.** *The 2-reachability closure for general graphs on n vertices can be computed in time $\mathcal{O}(n^\omega \log n)$.*

6 An application: computing all dominator trees

Let s be an arbitrary vertex of G. Recall the bridge decomposition $\mathcal{D}$ of (vertex-)dominator tree D and its tree T_v and root r_v from Section 4.2. We define the *edge-dominator tree* $\widetilde{D}$ of G with start vertex s, as the tree that results from D after contracting all vertices in each tree T_v into its root r_v. For any vertex v and edge $e = (x, y)$, e is contained in all paths in G from s to v if and only if (r_x, r_y) is in the path from s to r_v in $\widetilde{D}$. We denote by $\widetilde{d}(y)$ the parent of a vertex in $\widetilde{D}$. (Both y and $\widetilde{d}(y)$ are roots in $\mathcal{D}$.)

▶ **Theorem 17.** *We can compute all sources vertex- and edge-dominator trees from $G_R^{\rightsquigarrow 2e}$ in time $\mathcal{O}(n^2)$.*

We can preprocess each edge-dominator tree $\widetilde{D}$ in $\mathcal{O}(n)$ so as to answer ancestor-descendant relations in constant time [16]. We can also compute in $\mathcal{O}(n)$ time the number of descendants in D of every root r in $\mathcal{D}$. This allows us to answer various queries very efficiently:

- Given a pair of vertices s and t and an edge $e = (x, y)$, we can test if $G \setminus e$ contains a path from s to t in constant time. This is because e is contained in all paths from s to t

in G if and only if the following conditions hold: e is a bridge of flow graph G with start vertex s (i.e., $r_y = y$ and $\widetilde{d}(y) = r_x$) and y is an ancestor of r_t in $\widetilde{D}$.

- Similarly, given a vertex s and an edge $e = (x, y)$, we can report how many vertices become unreachable from s if we delete e from G. If e is a bridge of flow graph G with start vertex s, then this number is equal to the number of descendants of y in D. Hence, we find the edge whose removal disconnects the most pairs of vertices in time $\mathcal{O}(n^2)$.

By computing all vertex-dominator trees of G, we can answer analogous queries for vertex-separators. Moreover, we can answer efficiently queries regarding junctions. A vertex s is a *junction* of vertices u and v in G, if G contains a path from s to u and a path from s to v that are vertex-disjoint (i.e., s is the only vertex in common in these paths). Yuster [18] gave a $\mathcal{O}(n^\omega)$ algorithm to compute a single junction for every pair of vertices in a DAG. By having all dominator trees of a digraph G, we can also answer the following queries.

- Given vertices s, u and v, test if s is a junction of u and v. This is true if and only if u and v are descendants of distinct children of s in D. Hence, we perform this test in constant time.
- Similarly, we can report all junctions of a given a pair of vertices in $\mathcal{O}(n)$ time. Note that two vertices may have n junctions (e.g., in a complete graph).

Acknowledgments. We would like to thank Paolo Penna and Peter Widmayer for many useful discussions on the problem.

References

1. N. Alon, Z. Galil, O. Margalit, and M. Naor. Witnesses for boolean matrix multiplication and for shortest paths. In *Proc. of the 33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 417–426. IEEE, 1992. doi:10.1109/SFCS.1992.267748.
2. S. Alstrup, D. Harel, P. W. Lauridsen, and M. Thorup. Dominators in linear time. *SIAM Journal on Computing*, 28(6):2117–21132, 1999. doi:10.1137/S0097539797317263.
3. A. L. Buchsbaum, L. Georgiadis, H. Kaplan, A. Rogers, R. E. Tarjan, and J. R. Westbrook. Linear-time algorithms for dominators and other path-evaluation problems. *SIAM Journal on Computing*, 38(4):1533–1573, 2008. doi:10.1137/070693217.
4. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.
5. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.
6. M. J. Fischer and A. R. Meyer. Boolean matrix multiplication and transitive closure. In *Proc. of the 12th Annual Symposium on Switching and Automata Theory (SWAT)*, pages 129–131. IEEE, 1971. doi:10.1109/SWAT.1971.4.
7. W. Fraczak, L. Georgiadis, A. Miller, and R. E. Tarjan. Finding dominators via disjoint set union. *Journal of Discrete Algorithms*, 23:2–20, 2013. doi:10.1016/j.jda.2013.10.003.
8. L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-edge connectivity in directed graphs. *ACM Transactions on Algorithms*, 13(1):9:1–9:24, 2016. doi:10.1145/2968448.
9. L. Georgiadis, G. F. Italiano, and N. Parotsidis. Strong connectivity in directed graphs under failures, with applications. In *Proc. of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1880–1899, 2017. doi:10.1137/1.9781611974782.123.
10. G. F. Italiano, L. Laura, and F. Santaroni. Finding strong bridges and strong articulation points in linear time. *Theoretical Computer Science*, 447:74–84, 2012. doi:10.1016/j.tcs.2011.11.011.

11 R. Jin, Y. Xiang, N. Ruan, and H. Wang. Efficiently answering reachability queries on very large directed graphs. In *Proc. of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 595–608, New York, NY, USA, 2008. ACM. doi:10.1145/1376616.1376677.

12 V. King and G. Sagert. A fully dynamic algorithm for maintaining the transitive closure. *Journal of Computer and System Sciences*, 65(1):150–167, 2002. doi:10.1006/jcss.2002.1883.

13 F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, New York, NY, USA, 2014. ACM. doi:10.1145/2608628.2608664.

14 T. Lengauer and R. E. Tarjan. A Fast Algorithm for Finding Dominators in a Flowgraph. *ACM Transactions on Programming Languages and Systems*, 1(1):121–41, 1979. doi:10.1145/357062.357071.

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

16 R. E. Tarjan. Finding dominators in directed graphs. *SIAM Journal on Computing*, 3(1):62–89, 1974. doi:10.1137/0203006.

17 V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 887–898, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214056.

18 R. Yuster. All-pairs disjoint paths from a common ancestor in $\widetilde{O}(n^\omega)$ time. *Theoretical Computer Science*, 396(1):145–150, 2008. doi:10.1016/j.tcs.2008.01.032.

Edge-Orders

Lena Schlipf[1] **and Jens M. Schmidt**[*2]

1 **LG Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany**
`lena.schlipf@fernuni-hagen.de`
2 **Institut für Mathematik, TU Ilmenau, Ilmenau, Germany**
`jens.schmidt@tu-ilmenau.de`

Abstract

Canonical orderings and their relatives such as st-numberings have been used as a key tool in algorithmic graph theory for the last decades. Recently, a unifying link behind all these orders has been shown that links them to well-known graph decompositions into parts that have a prescribed vertex-connectivity.

Despite extensive interest in canonical orderings, no analogue of this unifying concept is known for edge-connectivity. In this paper, we establish such a concept named *edge-orders* and show how to compute *(1,1)-edge-orders* of 2-edge-connected graphs as well as *(2,1)-edge-orders* of 3-edge-connected graphs in linear time, respectively. While the former can be seen as the edge-variants of *st-numberings*, the latter are the edge-variants of *Mondshein sequences* and *non-separating ear decompositions*. The methods that we use for obtaining such edge-orders differ considerably in almost all details from the ones used for their vertex-counterparts, as different graph-theoretic constructions are used in the inductive proof and standard reductions from edge- to vertex-connectivity are bound to fail.

As a first application, we consider the famous *Edge-Independent Spanning Tree Conjecture*, which asserts that every k-edge-connected graph contains k rooted spanning trees that are pairwise edge-independent. We illustrate the impact of the above edge-orders by deducing algorithms that construct 2- and 3-edge independent spanning trees of 2- and 3-edge-connected graphs, the latter of which improves the best known running time from $O(n^2)$ to linear time.

1998 ACM Subject Classification G.2.2 Graph Theory, Graph Algorithms

Keywords and phrases edge-order, st-edge-order, canonical ordering, edge-independent spanning tree, Mondshein sequence, linear time

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.75

1 Introduction

Canonical orderings serve as a fundamental tool in various fields of algorithmic graph theory, see [2, 26] for a wealth of over 30 applications. Under this name, canonical orderings were published in 1988 for maximal planar graphs [8] and soon after generalized to 3-connected planar graphs [14]. Interestingly, it turned out only recently [26] that the well-known *non-separating ear decompositions* [6] are in fact strict generalizations of canonical orderings to arbitrary 3-connected graphs, and that this generalization was, independently, already known as *(2,1)-sequences* [19] in 1971 long before canonical orderings were even proposed (anticipating many of their later planar features).

Mondshein [19] characterized (2,1)-sequences, or *(2,1)-orders*, as we will call them, by decomposing a graph into 2-connected and connected parts. Indeed, the unifying link above

* This research was supported by the DFG grant SCHM 3186/1-1.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 75; pp. 75:1–75:14

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

Table 1 *Left*: (k,l)-orders of $(k+l)$-connected graphs known so far and the best-known running times for constructing them. *Right*: (k,l)-edge-orders of $(k+l)$-edge-connected graphs (this paper).

$k\backslash l$	1	2
1	*st*-numbering [9] $O(m)$	
2	Mondshein sequence [25] $O(m)$	Chain decomposition [7] $O(n^2m)$; if planar [21] $O(m)$
3	(3,1)-order for triangulations [4] $O(m)$	5-canonical decomposition for triangulations [20] $O(m)$
4		

$k\backslash l$	1
1	*st*-edge-numbering [1] $O(m)$ *(+in this paper)*
2	(2,1)-edge-order $O(m)$ *(in this paper)*
3	
4	

allows to describe any canonical ordering of a graph $G = (V, E)$ as a total order on V such that for certain i, the first i vertices induce a 2-connected graph and the remaining vertices induce a connected graph in G [26] (and hence, does not use any reference to planarity). The general concept behind canonical orderings is thus connectivity, with all of its implications for planarity, instead of planarity itself.

Several publications [20, 7, 4] extended this approach to (k,l)-*orders* with $(k,l) \neq (2,1)$. Such (k,l)-orders may be described canonically as total orders on V such that for certain i, the first i vertices induce a k-connected graph and the remaining vertices induce a l-connected graph (a related description for planar triangulations is given in [4]). We note that this is not a definition, as "certain i" has to be quantified for every particular (k,l). This is usually done in dependence of a graph decomposition, which tend to become more complex, as k or l grow: e.g. for $(2,1)$-orders, "certain i" is quantified by taking every vertex i that completes an ear with the predecessors of i in a fixed open ear decomposition of G.

Several relatives of (2,1)-orders fit into the context of (k,l)-orders: The well-known *st-numberings* and *st-orientations* are actually (1,1)-orders of 2-connected graphs, where i ranges over all vertices, the *chain decompositions* of [7] are (2,2)-orders of 4-connected graphs, and more orders on restricted graph classes such as planar graphs and triangulations are known (see Table 1 left).

The purpose of this paper is to extend this unifying view further to (k,l)-*edge-orders*, each of which can be described as a total order on E such that for certain i, the first i edges induce a k-edge-connected graph and the remaining edges induce a l-edge-connected graph. Despite the many known and heavily used vertex-orders above, these natural edge-variants do not seem to be well-studied. In fact, we are only aware of one technical report by Annexstein et al. [1], which deals with (1,1)-edge-orders (under the name *st-edge-orderings*). For the $(1,1)$-edge-order we present, i ranges over all edges except st; for the $(2,1)$-edge-order, i ranges over all edges that complete an ear with the predecessors of i in a fixed ear decomposition of G.

We show a simple algorithm how a (1,1)-edge-order can be computed and prove that it has running time $O(m)$. Our main contribution is then an algorithm that computes a (2,1)-edge-order of a 3-edge-connected graph in time $O(m)$ (see Table 1 right), of which the corresponding result for the vertex-counterpart took over 40 years.

Just like (2,1)-orders, which immediately led to improvements on the best-known running time for five applications [5, 26], (2,1)-edge-orders seem to be an important and useful tool for many graph algorithms. We give an application of them, which is related to the edge-independent spanning tree conjecture [13]: By using a (2,1)-edge-order, we show that three edge-independent spanning trees of 3-edge-connected graphs can be computed in time $O(m)$, improving the best-known running time $O(n^2)$ by Gopalan et al. [11].

We also considered the *3-edge-partition problem*, but surprisingly did not find an easy reduction to $(2,1)$-edge-orders. However, we note that this problem can be solved in linear time using existing algorithms: A 3-edge-partition can be computed by two linear-time reductions, first to the *vertex-subset tripartitioning problem* [28, Theorem 2b], and then [27] to the problem of computing a non-separating ear decomposition. It is also possible to find an alternative simple and direct linear-time reduction along the lines of [26, Application 5].

After giving preliminary facts on ear decompositions, we explain the linear-time algorithms for computing (1,1)- and (2,1)-edge-orders in Sections 3–5. Section 6 then shows algorithms for computing two and three edge-independent spanning trees.

1.1 Vertex-connectivity vs. edge-connectivity

In many cases, the vertex-variant of a connectivity problem is more challenging than its edge-variant, as the latter may be reduced to the former by taking its line-graph or by using the reduction from k-edge- to k-vertex-connectivity of Galil and Italiano [10]. From a top-level perspective, our (2,1)-edge-order algorithm follows the proof outline of its vertex-counterpart in [26]. Thus, it needs to be motivated that there is no obvious linear-time reduction to [26] that produces the results of this paper (of course there is a non-obvious reduction that just takes the algorithm of this paper and does not invoke [26] at all).

Clearly, a reduction to line-graphs is not possible, as this may involve a quadratic blow-up in the graph size and thus in the running time. Using the reduction of Galil-Italiano, we can reduce a 3-edge-connected graph G to a 3-connected graph G', and then compute a (2,1)-order of G' in linear time using [26]. However, it can be shown that there is no obvious way of transforming the (2,1)-order of G' back to a (2,1)-edge-order of G.

Another hint that such a reduction might be elusive is given by our application to edge-independent spanning trees. Despite extensive research, it is still not known how to reduce these to vertex-independent spanning trees (which may in turn be computed from a (2,1)-order [26]), not even for the corresponding existence results. In fact, an attempt trying to prove this turned out to be false [12]. If there was a reduction to (2,1)-orders, it would directly imply a reduction to vertex-independent spanning trees.

Hence, there is no obvious way of producing our results using old ones. Indeed, the different parts of our proof require substantially new ideas and non-trivial formalizations in comparison to [26]: Mader-sequences differ from the (BG)-sequences used in [26] (and, although they are not too far apart, it took a 27-page paper to show that the former can be computed in linear time as well [18]), the notions of non-separateness and $\overline{G_i}$ differ considerably, and, here, we need last-values in addition to just birth-values.

2 Preliminaries

We use standard graph-theoretic terminology and consider only graphs that are finite and undirected, but may contain parallel edges and self-loops. In particular, cycles may have length one or two. A separator of size one is called a *cut-vertex*. The *2-connected components* of a graph are its inclusion-wise maximal connected subgraphs having no cut-vertex. For $k \geq 1$, let a graph G be *k-edge-connected* if $n := |V| \geq 2$ and G has no edge-cut of size less than k.

▶ **Definition 1** ([15, 29]). An *ear decomposition* of a graph $G = (V, E)$ is a sequence $(P_0, P_1, \ldots, P_k)$ of subgraphs of G that partition E such that (i) P_0 is a cycle that is no self-loop and (ii) every P_i, $1 \leq i \leq k$, is either a path that intersects $P_0 \cup \cdots \cup P_{i-1}$ in its

endpoints or a cycle that intersects $P_0 \cup \cdots \cup P_{i-1}$ in a unique vertex q_i (which we call *endpoint* as well). Each P_i is called an *ear*. An ear is *short* if it is an edge and *long* otherwise.

▶ **Theorem 2** ([22]). *A graph is 2-edge-connected if and only if it has an ear decomposition.*

According to Whitney [29], every ear decomposition has exactly $m-n+1$ ears ($m := |E|$). For any i, let $G_i = (V_i, E_i) := P_0 \cup \cdots \cup P_i$ and $\overline{E_i} := E - E_i$. We denote the subgraph of G that is induced by $\overline{E_i}$ as $\overline{G_i} = (\overline{V_i}, \overline{E_i})$. Clearly, $\overline{G_j} \subset \overline{G_i}$ for every $i < j$. We note that this definition of $\overline{G_i}$ differs from the definition $\overline{G_i} := G - V_i$ that was used for (2,1)-vertex-orders [26], due to the weaker edge-connectivity assumption.

For any ear P_i, let $inner(P_i) := V(P_i) - G_{i-1}$ be the set of *inner vertices* of P_i (for P_0, every vertex is an inner vertex). Hence, for a cycle $P_i \neq P_0$, $inner(P_i) = V(P_i) - q_i$. Every vertex of G is an inner vertex of exactly one long ear, which implies that, in an ear decomposition, the inner vertex sets of the long ears partition V.

▶ **Definition 3.** Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an ear decomposition of G. For an edge e, let $birth_D(e)$ be the index i such that P_i contains e. For a vertex v, let $birth_D(v)$ be the index i such that P_i contains v as inner vertex and let $last_D(v)$ be the maximal index $birth(vw)$ over all neighbors w of v. Whenever D is clear from the context, we will omit the subscript D.

Thus, $P_{last(v)}$ is the last ear that contains v and, seen from another perspective, the first ear P_i such that $\overline{G_i}$ does not contain v. Clearly, a vertex v is contained in $\overline{G_i}$ if and only if $last(v) > i$.

3 The (1,1)-edge-order

Although (1,1)-edge-orders can be seen as edge-counterparts of *st*-numberings, they do not seem to be well-known. Let two edges be *neighbors* if they share a common vertex. Annexstein et al. gave essentially the following definition.

▶ **Definition 4** ([1]). Let $G = (V, E)$ be a graph with an edge st that is not a self-loop. A *(1,1)-edge-order through st* of G is a total order $<$ on the edge set $E - st$ such that $m \geq 2$,

- every edge e, except for one incident to s, has a neighbor e' with $e' < e$ and
- every edge e, except for one incident to t, has a neighbor e' with $e < e'$.

Hence, the two exceptional edges incident to s and t must be, respectively, the minimal and maximal edge of $E - st$ with respect to $<$. Clearly, if G has a (1,1)-edge-order through st, G is 2-edge-connected, as neither st nor any other edge can be a bridge of G (note that this requires $m \geq 2$). The converse statement was shown in [1, Prop. 4] using a special type of ear decompositions based on breadth-first-search (however, without giving details of the linear-time algorithm). Here, we aim for a simple and direct (unlike, e.g., reducing to (1,1)-orders via line-graphs) exposition of the underlying idea and show that *any* ear decomposition can be transformed to a (1,1)-edge-order in linear time.

We will use the *incremental list order-maintenance problem*, which maintains a total order subject to the operations of (i) *inserting* an element after a given element and (ii) *comparing* two distinct given elements by returning the one that is smaller in the order. Bender et al. [3] show a simple solution for an even more general problem with amortized constant time per operation; we will call this the *order data structure*.

▶ **Lemma 5.** *Let G be a 2-edge-connected graph with an edge st that is not a self-loop. Then a (1,1)-edge-order through st can be computed in time $O(m)$.*

Proof. We compute an ear decomposition D of G such that $st \in P_0$. This can be done in linear time by any text-book-algorithm; see [24] for a simple one. Let $<_0$ be the total order that orders the edges in $P_0 - st$ consecutively from s to t. Thus, every edge has a smaller and a larger neighbor, except for st and the two exceptional edges incident to s and t. Clearly, $<_0$ is a (1,1)-edge-order through st of the 2-edge-connected graph G_0. We extend $<_{i-1}$ iteratively to a (1,1)-edge-order $<_i$ of G_i by adding the next ear P_i of D; then $<_{m-n}$ gives the claim.

The order itself is stored in the *order data structure*. For every vertex x in G_{i-1}, let $min(x)$ be the smaller of its two incident edges in $P_{birth(x)}$ with respect to $<_{i-1}$ (for later arguments, define $max(x)$ analogously as the larger such edge); clearly, $min(x)$ and $max(x)$ can be computed in constant time while adding P_j. When adding the ear P_i with (not necessarily distinct) endpoints x and y, let e be the smallest edge in $\{min(x), min(y)\}$ with respect to $<_{i-1}$ (this needs amortized constant time by using at most one comparison of the data structure). Consider all edges of P_i in consecutive order starting with a neighbor of e. We obtain $<_i$ from $<_{i-1}$ by inserting these edges as one consecutive block immediately after the edge e (if P_i is a cycle with endpoint s the edges are insert in front of the other edges); this takes amortized time proportional to the length of P_i. Then the first edge of P_i has a smaller neighbor in $<_i$ while the last has a larger neighbor in $<_i$ (for cycles $P_i \neq P_0$, this exploits that q_i has another incident edge in G_{i-1} or the exceptional edge incident to s (or t) might change), which implies that $<_i$ is a (1,1)-edge-order. ◀

This (special) (1,1)-edge-order will allow for a very easy computation of two edge-independent spanning trees in Section 6 and serve as a building block for the computation of three such trees. If one wants to keep the root-paths in two edge-independent spanning trees short, a different (1,1)-edge-order [1] may be computed by maintaining $min(x)$ as the incident edge of x that is minimal in G_i in the above algorithm (this can be done efficiently by updating $min(x)$ whenever an ear with endpoint x is added). However, the latter order cannot be used for three edge-independent spanning trees.

4 The (2,1)-edge-order

We define (2,1)-orders as special ear decompositions.

▶ **Definition 6.** Let G be a graph with distinct edges rt and ru ($t = u$ is possible). A *(2,1)-edge-order* through rt and avoiding ru (see Figure 1) is an ear decomposition D of G such that

1. $rt \in P_0$,
2. $P_{m-n} = ru$, and ▷ i.e., the last ear is the short ear ru
3. for every $0 \leq i < m - n$, $\overline{G_i}$ contains $inner(P_i)$ and, if P_i is short, at least one endpoint of P_i.

Property 6.2 implies that $\overline{G_i}$ contains the vertices r and u for every $0 \leq i < m - n$. We call Property 6.3 the *non-separateness* of D. The non-separateness of D states that every inner vertex of a long ear P_i has an incident edge in G that is in $\overline{G_i}$, and that every short ear P_i (seen as edge) has a neighbor in $\overline{G_i}$. The name refers to the following helpful property.

▶ **Lemma 7.** *Let D be a (2,1)-edge-order. Then, for every $0 \leq i < m - n$, $\overline{G_i}$ is connected.*

Proof. Consider any $i < m - n$ and let e be any edge in $\overline{G_i}$. By Property 6.2, $r \in \overline{G_i}$. We show that $\overline{G_i}$ contains a path from one of the endpoints of e to r. This gives the claim, as $\overline{G_i}$ is an edge-induced graph and therefore does not contain isolated vertices.

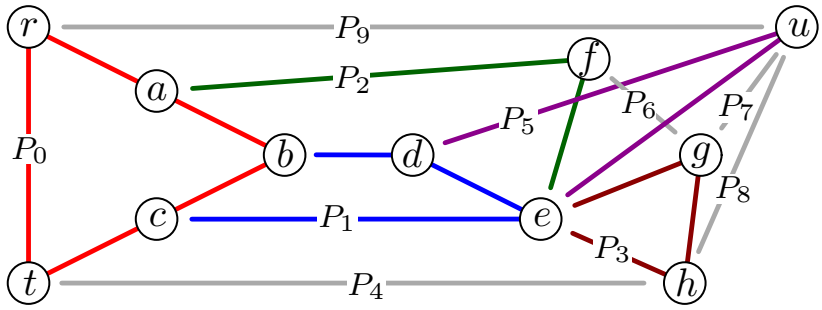


Figure 1 A (2,1)-edge-order of a 3-edge connected graph.

Let P_j be the unique ear that contains e. If P_j is short, $P_j = e$ and e has a neighbor in $\overline{G_j}$ due to the non-separateness of D. If P_j is long, at least one endpoint of e must be an inner vertex of P_j and e has a neighbor in $\overline{G_j}$ for the same reason. Hence, in both cases we find a neighbor that is contained in an ear P_k with $k > j$. By applying induction on the indices of these ears, we find a path that starts with an endpoint of e and ends with the only edge left in $\overline{G_{m-n-1}}$, namely ru. ◀

Next, we show that the existence of a (2,1)-edge-order proves the graph to be 3-edge-connected.

▶ **Lemma 8.** *If G has a (2,1)-edge-order, G is 3-edge-connected.*

Proof. Let D be a (2,1)-edge-order through rt and avoiding ru. Consider any vertex v of G. By transitivity of edge-connectivity, it suffices to show that G contains three edge-disjoint paths between v and r. Let P_i be the ear that contains v as inner vertex. In particular $i < m - n$, as P_i is long. Then G_i has an ear decomposition and, due to Theorem 2, contains two edge-disjoint paths between v and r. By Properties 6.2+3, $\overline{G_i}$ contains v and r. According to Lemma 7, $\overline{G_i}$ is connected. Thus, $\overline{G_i}$ contains a third path between v and r, which is edge-disjoint from the first two, as G_i and $\overline{G_i}$ are edge-disjoint. ◀

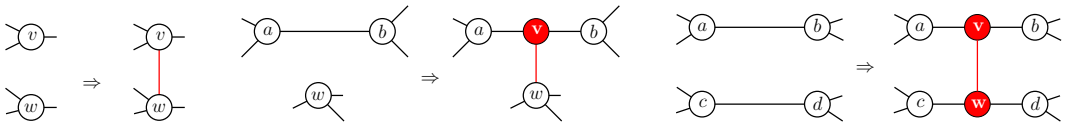
Let G have a (2,1)-edge-order. Then Lemma 8 implies $\delta(G) \geq 3$. This in turn gives that, for every vertex v, $P_{last(v)}$ is not the first ear that contains v, which implies that $P_{last(v)}$ must have v as endpoint. In particular, if vw is an edge and $last(v) = last(w) = birth(vw)$, $P_{birth(vw)}$ is the short ear vw and, according to the non-separateness of D, we have $i = m-n$, which implies $vw = ru$.

▶ **Lemma 9.** *For any vertex v, $P_{last(v)}$ has v as an endpoint. For any edge vw satisfying $last(v) = last(w) = birth(vw)$, $vw = ru$.*

The converse of Lemma 8 is also true: If G is 3-edge-connected, G has a (2,1)-edge-order. This gives a full characterization of 3-edge-connected graphs; however, proving the latter direction is more involved than Lemma 8. In the next section, we will prove the stronger statement that such a (2,1)-edge-order does not only exist but can actually be computed efficiently.

5 Computing a (2,1)-edge-order

At the heart of our algorithm is the following classical construction of 3-edge-connected graphs due to Mader.



(a) vertex-vertex-addition: $v = w$ is allowed.

(b) edge-vertex-addition: $w \in \{a, b\}$ is allowed.

(c) edge-edge-addition: $a, b \in \{c, d\}$ is allowed.

Figure 2 Mader-operations.

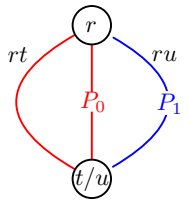


Figure 3 A (2,1)-edge-order of K_2^3 through rt and avoiding ru.

▶ **Definition 10.** The following operations on graphs are called *Mader-operations* (see Figure 2).

(a) *vertex-vertex-addition*: Add an edge between the not necessarily distinct vertices v and w (possibly a parallel edge or, if $v = w$, a self-loop).

(b) *edge-vertex-addition*: Subdivide an edge ab with a vertex v and add the edge vw for a vertex w.

(c) *edge-edge-addition*: Subdivide two distinct edges ab and cd with vertices v and w, respectively, and add the edge vw.

The edge vw is called the *added edge* of the Mader-operation. Let K_2^3 be the graph that consists of exactly two vertices and three parallel edges.

▶ **Theorem 11** ([16]). *A graph G is 3-edge-connected if and only if G can be constructed from K_2^3 using Mader-operations.*

According to Theorem 11, applying Mader-operations on 3-edge-connected graphs preserves 3-edge-connectivity. We will call a sequence of Mader-operations that constructs a 3-edge-connected graph a *Mader-sequence*. It has been shown that a Mader-sequence can be computed efficiently.

▶ **Theorem 12** ([18, Thm. 4]). *A Mader-sequence of a 3-edge-connected graph can be computed in time $O(n + m)$.*

Our algorithm for computing a (2,1)-edge-order works as follows. Assume we want a (2,1)-edge-order of G through $r\bar{t}$ and avoiding $r\bar{u}$. We first compute a suitable Mader-sequence of G using Theorem 12 and start with a (2,1)-edge-order of its first graph K_2^3. This (2,1)-edge-order is easy to find (see Figure 3). The crucial part of the algorithm is then to iteratively modify the given (2,1)-edge-order to a (2,1)-edge-order of the next graph in the sequence efficiently.

There are several technical difficulties to master. First, the edges $r\bar{t}$ and $r\bar{u}$ may be contained in different 2-connected components A' and B' (implying that r is a cut-vertex). As this would raise problems in the computation of the initial K_2^3 later, we perform in such a case the following reduction in advance. Let $\overline{A}$ be the connected component of $G \setminus \{r\}$

containing $\bar{t}$, $A := G[V(\overline{A}) \cup \{r\}]$ and $B := G \setminus V(\overline{A})$ (note that r may still be a cut-vertex of B). Since r is a cut-vertex of G, A and B are still 3-edge-connected. We compute a (2,1)-edge-order D_A of A avoiding $r\bar{t}$ through an arbitrary edge $ru_A \in A' \setminus \{r\bar{t}\}$, and a (2,1)-edge-order D_B of B avoiding an arbitrary edge $rt_B \in B' \setminus \{r\overline{u}\}$ through $r\overline{u}$. Then concatenating D_A with D_B gives a (2,1)-edge-order of G. Hence, we assume from now on that $r\bar{t}$ and $r\overline{u}$ are in the same 2-connected component of the input graph G.

Second, the edge $r\bar{t}$ (and analogously $r\overline{u}$) of G is not necessarily contained in the previous graph of the Mader-sequence, as it may have been created by a Mader-operation that subdivided a previous edge rt with the new vertex $\bar{t}$ (a more general view on this dynamics follows from the bijection between the graphs H of the Mader-sequence and H-subdivisions that are contained in G as subgraphs [18, Thm.+Cor. 1]; we refer to [23, Sections 2.3 and 4] for details of this bijection). In such cases, we take t as *replacement* vertex for $\bar{t}$ (and likewise u for $\overline{u}$) in the previous graph, and iterate this procedure to obtain replacement vertices for t and u in the graph before that previous graph, and so forth. This way, the replacement vertices t and u in any graph of the Mader-sequence containing r are neighbors of r.

Now a special Mader-sequence is used to harness the dynamics of the vertices r, t and u: Choose a DFS-tree of G with root r such that $r\bar{t}$ and $r\overline{u}$ are backedges (this is possible, since r has degree at least three) and compute a Mader-sequence of this DFS-tree that contains these two edges in its initial K_2^3 (this is possible, since $r\bar{t}$ and $r\overline{u}$ are in the same 2-connected component of G). This way the K_2^3 consists of the two vertices r and $t = u$ by the construction of [18, p. 6], and thus all graphs in the Mader-sequence contain r (and t and u are always neighbors of r). The vertices $\bar{t}$ and $\overline{u}$ are not present in this initial K_2^3 unless they are identical to $t = u$ (they are however contained in the two paths from r to $t = u$ of the K_2^3-subdivision the bijection maps to). For every graph in the Mader-sequence, we will compute a (2,1)-edge-order through rt and avoiding ru using the previous (2,1)-edge-order (which depends on the previous and possibly different replacement vertices); then the choice of t and u ensures that the final (2,1)-edge-order of G is indeed through $r\bar{t}$ and avoids $r\overline{u}$, as desired.

Thus, consider a graph G of the above Mader-sequence for which we know a (2,1)-edge-order D and let G' be the next graph in that sequence. Then G' is only one Mader-operation away and we aim for an efficient modification of D into a (2,1)-edge-order D' of G'. We will prove that there is always a modification that is local in the sense that the only ears that are modified are "near" the added edge of the Mader-operation.

▶ **Lemma 13.** *Let $D = (P_0, P_1, \ldots, P_{m-n})$ be a (2,1)-edge-order of a 3-edge-connected graph G through rt and avoiding ru for replacement vertices t and u. Let G' be obtained from G by applying one Mader-operation Γ and let t' and u' be the replacement vertices of G'. Then a (2,1)-edge-order D' of G' through rt' avoiding ru' can be computed from D using only constantly many amortized constant-time modifications.*

Lemma 13 is our main technical contribution and we split its proof into the following three sections. First, we introduce the operations *leg*, *belly* and *head* in order to combine several cases that can be handled similarly for the different types of Γ. Second, we show how to modify D to D' and, third, we discuss computational issues.

For all three sections, let vw be the added edge of Γ such that v subdivides the edge $ab \in E(G)$ and w subdivides $cd \in E(G)$ (if applicable). Thus, the vertex t' in G' is either t, v or w, and the vertex u' in G' is either u, v or w (hence, $t'r$ and ru' will never be self-loops). In all three sections, *birth* and *last* will always refer to D, unless stated otherwise.

Let $P_i \neq P_0$ be an ear with a given orientation and let x be a vertex in P_i. If P_i is a path, we define $P_i[, x]$ and $P_i[x,]$ as the *maximal subpaths* of P_i that end and start at x,

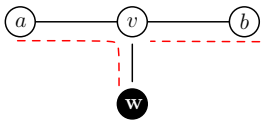


Figure 4 The result of operation *leg* (dashed lines), black vertices are in $G_{birth(ab)-1}$.

respectively; if P_i is a cycle, we take the same definition with the additional restriction that $P_i[, x]$ starts at q_i and $P_i[x,]$ ends at q_i. Occasionally, the orientation of P_i will not matter; if none is given, an arbitrary orientation can be taken. For paths A and B, let $A + B$ be the *concatenation* of A and B.

5.1 Legs, bellies and heads

While the operations *leg* and *belly* are inspired by the ones in [26], the operation *head* is new. All three operations will show for some special cases how D can be modified to a (2,1)-edge-order D'. A complete description for all cases (using these operations) will be given in the next section.

5.1.1 Legs

Let Γ be either an edge-vertex-addition such that $ab \neq ru$ and $last(w) < birth(ab)$ or an edge-edge-addition such that $ab \neq ru$ and $birth(cd) < birth(ab)$. If $P_{birth(ab)}$ is long, at least one of a and b is an inner vertex, say w.l.o.g. b. Otherwise, $P_{birth(ab)} = ab$ is short and, as D is non-separating, at least one of a and b, say w.l.o.g. b, has an incident edge in $\overline{G_{birth(ab)}}$ (note that this requires $ab \neq ru$). In both cases, orient $P_{birth(ab)}$ from a to b. The operation *leg* constructs D' from D by replacing the ear $P_{birth(ab)}$ of D by the two consecutive ears $P_{birth(ab)}[, a] + av + vw$ and $vb + P_{birth(ab)}[b,]$ in that order and, if Γ is an edge-edge-addition, additionally subdividing the edge cd in $P_{birth(cd)}$ with w (see Figure 4). Note that this definition is well-defined also for cycles $P_{birth(ab)}$, including self-loops.

We omit the proof that D' is a (2,1)-edge-order through rt' avoiding ru'.

5.1.2 Bellies

Let Γ be either an edge-vertex-addition such that $last(w) = birth(ab)$ and $w \notin \{a, b\}$ or an edge-edge-addition such that $birth(cd) = birth(ab)$ (note that $c, d \in \{a, b\}$ is allowed.) Consider the shortest path in $P_{birth(ab)}$ from an endpoint to one of the vertices $\{a, b\}$, say w.l.o.g. b, such that w is contained in this path. We orient $P_{birth(ab)}$ from a to b. $P_{birth(ab)}$ is a long ear with b as inner vertex. If Γ is an edge-edge-addition, one of the vertices $\{c, d\}$, say w.l.o.g. c, is contained in $P_{birth(ab)}[, w]$.

If $birth(ab) > 0$, the operation *belly* constructs D' from D by replacing the ear $P_{birth(ab)}$ of D by the two consecutive ears $P_{birth(ab)}[, a] + av + vw + P_{birth(ab)}[w,]$ and $vb + P_{birth(ab)}[b, w]$ in that order (if edge-vertex-addition) and by the two consecutive ears $P_{birth(ab)}[, a] + av + vw + wd + P_{birth(ab)}[d,]$ and $vb + P_{birth(ab)}[b, c] + cw$ (if edge-edge-addition), see Figure 5. Note that this definition is well-defined also if $P_{birth(ab)}$ is a cycle. If $birth(ab) = 0$, the vertices v and w cut P_0 in two distinct paths $P_{0,1}$ and $P_{0,2}$ having endpoints v and w. Let $P_{0,1}$ be the path containing r. Then the operation *belly* constructs D' from D by replacing the ear $P_{birth(ab)}$ of D by the two consecutive ears $P_{0,1} + vw$ and $P_{0,2}$ in this order. If $rt \in \{ab, cd\}$, then either $v = t'$ or $w = t'$, respectively.

We omit the proof that D' is a (2,1)-edge-order through rt' avoiding ru'.

Figure 5 The result of the operation *belly* (dashed lines).

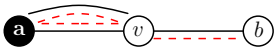


Figure 6 The dashed lines show the result of the operation *head*.

5.1.3 Heads

Let Γ be an edge-vertex-addition such that $w \in \{a, b\}$, $last(a) = birth(ab)$ and, if $ab = ru$, then $r \neq a$. W.l.o.g. let $w = a$. Then a is an endpoint of $P_{birth(ab)}$ ($P_{birth(ab)}$ cannot be a self-loop, as $last(a) = birth(ab)$). We orient $P_{birth(ab)}$ from a to b. The operation *head* constructs D' from D by replacing the ear $P_{birth(ab)}$ of D by the two consecutive ears $av + va$ and $vb + P_{birth(ab)}[b,]$ in that order (see Figure 6). Note that this definition is well-defined also for cycles $P_{birth(ab)}$.

We omit the proof that D' is a (2,1)-edge-order through rt' avoiding ru'.

5.2 Modifying D to D'

We will now show how to obtain a (2,1)-edge-order D' through rt' avoiding ru' from D. By symmetry, assume w.l.o.g. that $birth(ab) \geq birth(cd)$. Note that applying the operations *belly*, *leg* and *head* preserves all properties of a $(2,1)$-edge-order. Recall that, for every subdivision the Mader-sequences does on rt or ru, respectively, the subdividing vertex is t' or u', as explained after Figure 3. We have the following case distinctions:

1. **Γ is a vertex-vertex-addition.** (See Figure 2a.)
   - **(a)** vw is a self-loop at v ($v = w$): Obtain D' from D by adding the new short ear vv directly after the ear $P_{last(v)-1}$. This ensures that the new ear is non-separating.
   - **(b)** $v \neq w$ and $vw \neq \{rt, ru\}$: If $last(v) \leq last(w)$, D' is obtained from D by adding the new short ear vw directly after the ear $P_{last(w)-1}$, ensuring that the new ear is non-separating. If $last(v) > last(w)$, the new short ear vw is added directly after the ear $P_{last(v)-1}$.
   - **(c)** $vw = rt$ (the added edge is a parallel edge): the Mader-sequence gives us the information whether rt is rt' or the new added edge is rt'. If $rt = rt'$ then add the new edge immediately after the ear $P_{last(t)-1}$. Otherwise obtain D' from D by replacing rt with rt' in P_0 and adding the old edge rt as an short ear immediately after the ear $P_{last(t)-1}$.
   - **(d)** $vw = ru$ (the added edge is a parallel edge): the Mader-sequence gives us the information whether ru is ru' or the new added edge is ru'. Depending on this information, obtain D' from D by either adding the new edge directly before or directly after the last ear of D.
5. **Γ is an edge-vertex-addition.** (See Figure 2b.)
   - **(a)** $birth(ab) < last(w)$: Obtain D' from D by adding the new short ear vw directly after the ear $P_{last(w)-1}$ and subdivide the ear $P_{birth(ab)}$ with v. This operation is also well-defined when $P_{birth(ab)}$ is a cycle or self-loop. Also, the new ear is non-separating and, since v is incident to w, the ear $P_{birth(ab)}$ remains non-separating.

- **(b)** $last(w) < birth(ab)$ and $ab \neq ru$: Apply *leg*
- **(c)** $birth(ab) = last(w)$ and $w \notin \{a, b\}$: Apply *belly*.
- **(d)** $birth(ab) = last(w)$ and $w \in \{a, b\}$; if $ab = ru$, then $r \neq w$: Apply *head*.
- **(e)** $ab = ru$ and if $birth(ab) = last(w)$ and $w \in \{a, b\}$ then $r = w$: Obtain D' from D by replacing the ear ru by the two consecutive ears $wv + vu$ and rv.

6. **Γ is an edge-edge-addition.** (See Figure 2c.)
   - **(a)** $birth(ab) = birth(cd)$: Apply *belly*.
   - **(b)** $birth(ab) > birth(cd)$ and $ab \neq ru$: Apply *leg*.
   - **(c)** $ab = ru$: Let w.l.o.g. $r = a$. Obtain D' from D by replacing the last ear of D by the two consecutive ears $bv + vw$ and rv in this order.

In all cases, D' is clearly an ear decomposition. Properties 6.1–3 are satisfied due to the given case distinction and the mentioned properties. Hence, D' is a $(2, 1)$-edge-order through rt' avoiding ru'.

There are several subtleties in sorting out the computational complexity of this approach, mostly raised by the question how fast we can compute one of the above cases in which we are in. The proof of the linear runtime is omitted due to space constraints.

▶ **Theorem 14.** *Given edges tr and ru of a 3-edge-connected graph G, a (2,1)-edge-order D of G through tr and avoiding ru can be computed in time $O(m)$.*

The proposed algorithms for (1,1)-edge-orders and (2,1)-edge-orders (as well as the computation of edge-independent spanning trees in the next section) are *certifying* in the sense of [17]: For (1,1)-edge-orders through st, it suffices to check that every edge $e \neq st$ has indeed a smaller and larger neighboring edge. For (2,1)-edge-orders, it suffices to check in linear time that D is an ear decomposition of G and that D satisfies Properties 6.1–3.

6 Edge-Independent Spanning Trees

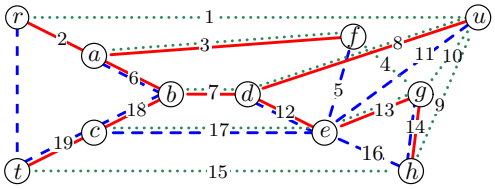
Let k spanning trees of a graph be *edge-independent* if they all have the same root vertex r and, for every vertex $x \neq r$, the paths from x to r in the k spanning trees are edge disjoint. The following conjecture was stated 1988 by Itai and Rodeh.

▶ **Conjecture** (Edge-Independent Spanning Tree Conjecture [13]). *Every k-edge-connected graph contains k edge-independent spanning trees.*

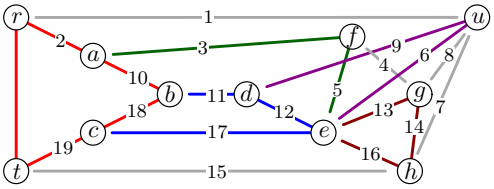
The conjecture has been proven constructively for $k \leq 2$ [13] and $k = 3$ [11] with running times $O(m)$ and $O(n^2)$, respectively, for computing the corresponding edge-independent spanning trees. For every $k \geq 4$, the conjecture is open. We first give a short description of an algorithm for $k = 2$ and then show the first linear-time algorithm for $k = 3$.

For $k = 2$, compute the (1,1)-edge-order $<$ through tr using Lemma 5. The first tree T_1 consists of the edges $min(x)$ for all vertices $x \neq r$ (as defined in Lemma 5), while the second tree T_2 consists of tr and the edges $max(x)$ for all vertices $x \notin \{r, t\}$. Then T_1 and T_2 are spanning, as no edge can be taken twice, and edge-independent, as, from every vertex x, the path of smaller edges to r obtained by iteratively applying $min()$ must be edge-disjoint from the path of larger edges to r.

For $k = 3$, choose any vertex r and two distinct edges tr and ru in the 3-edge-connected graph G. Compute a (2,1)-edge-order D through tr and avoiding ru in time $O(m)$ using Theorem 14. For every vertex $x \in V$, the idea is now to find *two* edge-disjoint paths from x to r in $G_{birth(x)}$ (after all, $G_{birth(x)}$ is 2-edge-connected and thus contains a (1,1)-edge-order)



(a) A consistent order $<$ and the resulting three edge-independent spanning trees.



(b) Although $<$ is a (1,1)-edge-order for every G_i, $0 \leq i \leq m-n$, $<$ is not consistent: Any down-consistent tree contains the root-paths $12, 11, 10, 2$ in G_2 and $6, 5, 3, 2$ in G_5, which implies a cycle.

Figure 7 (1,1)-edge-orders that are consistent and not consistent to the (2,1)-edge-order of Figure 1.

and a *third* path from x to r in $\overline{G_{birth(x)}}$ using the non-separateness of D. The subtle part is to make this idea precise: We have to construct the first tree T_1 in such a consistent way that the paths of smaller edges from x to r for *all* vertices $x \in V$ are contained in T_1 (and the same for T_2 and paths of larger edges).

For a (1,1)-edge-order $<$ through tr of G, let a spanning tree $T_1 \subseteq G$ be *down-consistent* to a given (2,1)-edge-order through tr if (a) every path in T_1 to r is strictly decreasing in $<$ and (b) for every $0 \leq i \leq m-n$, $T_1 \cap G_i$ is a spanning tree of G_i (analogously, *up-consistent* spanning trees T_2 of $G - r$ are defined by strictly increasing paths to t). Now let a (1,1)-edge-order be *consistent* to a given (2,1)-edge-order D' if G contains r-rooted spanning trees T_1 and T_2 that are down- and up-consistent to D', respectively. By the very same argument as used for $k = 2$, T_1 and $T_2 + tr$ are edge-independent and, in addition, do not use any edge of $\overline{G_{birth(x)}}$ for any $x \in V$.

In fact, the special (1,1)-edge-order that is computed by Lemma 5 is consistent to D: There, the trees T_1 and T_2 consist of the edges $min(x)$ and $max(x)$ for $x \in V$, which makes T_1 down-consistent and $T_2 + tr$ up-consistent to D (see Figure 7a). We note that a simpler definition of consistent as used for the vertex-variant [6], i.e., as *orders that remain (1,1)-edge-orders for all subgraphs* G_i, $0 \leq i \leq m-n$, does not suffice here (see Figure 7b).

It remains to construct the third edge-independent spanning tree. For every edge $e \neq ru$ of G, we compute a pointer to an arbitrary neighboring edge e' in $\overline{G_{birth(e)}}$. This edge e' exists, as D is non-separating, and satisfies $birth(e') > birth(e)$. Similarly, for every vertex $x \in V - r - u$, we compute a pointer to an incident edge e' of x with $birth(e') > birth(x)$. Both computations take linear total time by comparing *birth* values. The third edge-independent spanning tree is then the union of ur and the u-rooted spanning tree of $G - r$ that interprets the pointers as parent edges. Hence, we obtain the following theorem.

▶ **Theorem 15.** *Given the two edges rt and ru of a 3-edge-connected graph G, three edge-independent spanning trees of G rooted at r (such that two of them contain rt and ru as unique root edges, respectively) can be computed in time $O(m)$.*

Similarly as for the more general (2,1)-edge-orders, one could be interested why the reduction from k-edge- to k-vertex-connectivity by Galil and Italiano [10] does not give edge-independent spanning trees from their vertex-counterparts; we omit the argument due to space constraints.

References

1. Fred Annexstein, Ken Berman, and Ram Swaminathan. Independent spanning trees with small stretch factors. Technical Report 96-13, DIMACS, June 1996.
2. M. Badent, U. Brandes, and S. Cornelsen. More canonical ordering. *Journal of Graph Algorithms and Applications*, 15(1):97–126, 2011.
3. M. A. Bender, R. Cole, E. D. Demaine, M. Farach-Colton, and J. Zito. Two simplified algorithms for maintaining order in a list. In *Proceedings of the 10th European Symposium on Algorithms (ESA'02)*, pages 152–164, 2002. doi:10.1007/3-540-45749-6_17.
4. T. Biedl and M. Derka. The (3,1)-ordering for 4-connected planar triangulations. *Journal of Graph Algorithms and Applications*, 20(2):347–362, 2016.
5. T. Biedl and J. M. Schmidt. Small-area orthogonal drawings of 3-connected graphs. In *Proceedings of the 23rd International Symposium on Graph Drawing (GD'15)*, pages 153–165, 2015. doi:10.1007/978-3-319-27261-0_13.
6. J. Cheriyan and S. N. Maheshwari. Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs. *Journal of Algorithms*, 9(4):507–537, 1988.
7. S. Curran, O. Lee, and X. Yu. Chain decompositions of 4-connected graphs. *SIAM J. Discrete Math.*, 19(4):848–880, 2005. doi:10.1137/S0895480103434592.
8. H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting fary embeddings of planar graphs. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88)*, pages 426–433, 1988. doi:10.1145/62212.62254.
9. S. Even and R. E. Tarjan. Computing an st-Numbering. *Theor. Comput. Sci.*, 2(3):339–344, 1976.
10. Z. Galil and G. F. Italiano. Reducing edge connectivity to vertex connectivity. *SIGACT News*, 22(1):57–61, 1991. doi:10.1145/122413.122416.
11. A. Gopalan and S. Ramasubramanian. On constructing three edge independent spanning trees. Manuscript (see http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.406.7119), March 2011.
12. A. Gopalan and S. Ramasubramanian. A counterexample for the proof of implication conjecture on independent spanning trees. *Information Processing Letters*, 113(14-16):522–526, 2013. doi:10.1016/j.ipl.2013.04.008.
13. A. Itai and M. Rodeh. The multi-tree approach to reliability in distributed networks. *Information and Computation*, 79:43–59, 1988.
14. G. Kant. Drawing planar graphs using the lmc-ordering. In *Proceedings of the 33th Annual Symposium on Foundations of Computer Science (FOCS'92)*, pages 101–110, 1992. doi:10.1109/SFCS.1992.267814.
15. L. Lovász. Computing ears and branchings in parallel. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS'85)*, pages 464–467, 1985. doi:10.1109/SFCS.1985.16.
16. W. Mader. A reduction method for edge-connectivity in graphs. In B. Bollobás, editor, *Advances in Graph Theory*, volume 3 of *Annals of Discrete Mathematics*, pages 145–164. North-Holland, 1978. doi:10.1016/S0167-5060(08)70504-1.
17. R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011. doi:DOI:10.1016/j.cosrev.2010.09.009.
18. K. Mehlhorn, A. Neumann, and J. M. Schmidt. Certifying 3-edge-connectivity. *Algorithmica*, 77(2):309–335, 2017. doi:10.1007/s00453-015-0075-x.
19. L. F. Mondshein. *Combinatorial Ordering and the Geometric Embedding of Graphs*. PhD thesis, M.I.T. Lincoln Laboratory / Harvard University, 1971. Technical Report available at http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0732882.

20 S. Nagai and S. Nakano. A linear-time algorithm to find independent spanning trees in maximal planar graphs. In *26th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'00)*, pages 290–301, 2000. doi:10.1007/3-540-40064-8_27.

21 S. Nakano, Md. S. Rahman, and T. Nishizeki. A linear-time algorithm for four-partitioning four-connected planar graphs. *Inf. Process. Lett.*, 62(6):315–322, 1997. doi:10.1016/S0020-0190(97)00083-5.

22 H. E. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939. URL: http://www.jstor.org/stable/2303897.

23 J. M. Schmidt. Construction sequences and certifying 3-connectedness. In *Proceedings of the 27th Symposium on Theoretical Aspects of Computer Science (STACS'10)*, pages 633–644, 2010. URL: http://arxiv.org/abs/0912.2561, doi:10.4230/LIPIcs.STACS.2010.2491.

24 J. M. Schmidt. A simple test on 2-vertex- and 2-edge-connectivity. *Information Processing Letters*, 113(7):241–244, 2013. doi:10.1016/j.ipl.2013.01.016.

25 J. M. Schmidt. The Mondshein sequence. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP'14)*, pages 967–978, 2014. doi:10.1007/978-3-662-43948-7_80.

26 J. M. Schmidt. Mondshein sequences (a.k.a. (2,1)-orders). *SIAM Journal on Computing*, 45(6):1985–2003, 2016. doi:10.1137/15M1030030.

27 K. Wada and K. Kawaguchi. Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs. In *19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'93)*, pages 132–143, 1993. doi:10.1007/3-540-57899-4_47.

28 K. Wada, A. Takaki, and K. Kawaguchi. Efficient algorithms for a mixed k-partition problem of graphs without specifying bases. *Theoretical Computer Science*, 201:233–248, 1998.

29 H. Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34(1):339–362, 1932.

Relaxations of Graph Isomorphism*†

Laura Mančinska[1], David E. Roberson[2], Robert Šámal[3], Simone Severini[4], and Antonios Varvitsiotis[5]

1 **University of Bristol, Bristol, UK**
laura.mancinska@bristol.ac.uk
2 **University College London, London, UK**
davideroberson@gmail.com
3 **Charles University, Prague, Czech Republic**
samal@iuuk.mff.cuni.cz
4 **University College London, London, UK; and Shanghai Jiao-Tong University, Shanghai, China**
s.severini@ucl.ac.uk
5 **Nanyang Technological University and Centre for Quantum Technologies, Singapore**
avarvits@gmail.com

Abstract

We introduce a nonlocal game that captures and extends the notion of graph isomorphism. This game can be won in the classical case if and only if the two input graphs are isomorphic. Thus, by considering quantum strategies we are able to define the notion of quantum isomorphism. We also consider the case of more general non-signalling strategies, and show that such a strategy exists if and only if the graphs are fractionally isomorphic. We prove several necessary conditions for quantum isomorphism, including cospectrality, and provide a construction for producing pairs of non-isomorphic graphs that are quantum isomorphic.

We then show that both classical and quantum isomorphism can be reformulated as feasibility programs over the completely positive and completely positive semidefinite cones respectively. This leads us to considering relaxations of (quantum) isomorphism arrived at by relaxing the cone to either the doubly nonnegative (DNN) or positive semidefinite (PSD) cones. We show that DNN-isomorphism is equivalent to the previous defined notion of graph equivalence, a polynomial-time decidable relation that is related to coherent algebras. We also show that PSD-isomorphism implies several types of cospectrality, and that it is equivalent to cospectrality for connected 1-walk-regular graphs. Finally, we show that all of the above mentioned relations form a strict hierarchy of weaker and weaker relations, with non-singalling/fractional isomorphism being the weakest. The techniques used are an interesting mix of algebra, combinatorics, and quantum information.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases graph isomorphism, quantum information, semidefinite programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.76

* The full version of this work, including all of the proofs, is given in [3] (https://arxiv.org/abs/1611.09837) and [16].

† LM is supported by UK EPSRC under grant EP/L021005/1. DR is supported by Cambridge Quantum Computing Ltd. and the Engineering and Physical Sciences Research Council of the United Kingdom (EPSRC), as well as Simone Severini and Fernando Brandao. RS is partially supported by grant GA ČR P202-12-G061 and by grant LL1201 ERC CZ of the Czech Ministry of Education, Youth and Sports. SS is supported by the Royal Society, the EPSRC, and the National Natural Science Foundation of China (NSFC). AV is supported in part by the Singapore National Research Foundation under NRF RF Award No. NRF-NRFF2013-13. Part of this work was done while DR, and SS were visiting the Simons Institute for the Theory of Computing.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 76; pp. 76:1–76:14

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

1 Introduction

Given graphs G and H, an *isomorphism* from G to H is a bijection $\varphi : V(G) \to V(H)$ such that $\varphi(g)$ is adjacent to $\varphi(g')$ if and only if g is adjacent to g'. When such an isomorphism exists, we say that G and H are *isomorphic* and write $G \cong H$. The notion of isomorphism is central to a broad area of mathematical research encompassing algebraic and structural graph theory, but also combinatorial optimization, parameterized complexity, and logic. The *graph isomorphism (GI) problem* consists of deciding whether two graphs are isomorphic. It is a question with fundamental practical interest due to the number of problems that can be reduced to it. Additionally, the GI problem has a central role in theoretical computer science as it is one of the few naturally defined problems in NP which is not known to be polynomial-time solvable or NP-complete. While there is a deterministic quasipolynomial algorithm for the GI problem [5], regardless of its worst case behavior, the problem can be solved with reasonable efficiency in practice (*e.g.* see [17]). In relation to the context of this paper, it is valuable to notice that the discussion around graph isomorphism has branched into the analysis of many equivalence relations that form hierarchical structures. Prominent instances are, for example, cospectrality, fractional isomorphism, *etc.* [4, 12, 26].

In this work we introduce the graph isomorphism game, which is played by non-communicating players and allows us to capture and extend the notion of graph isomorphism. We investigate three classes of strategies for this game: classical, quantum, and non-signalling. In the classical case, players can win the (G, H)-isomorphism game with certainty if and only if $G \cong H$. This motivates the definition of graphs G and H being quantum, or non-signalling, isomorphic if there exists a perfect quantum, resp. non-signalling, strategy for this game. These two relations are denoted by $G \cong_q H$ and $G \cong_{ns} H$ respectively. We are able to prove two algebraic characterizations of quantum isomorphism, one of which implies that quantum isomorphic graphs are cospectral. We also show that non-signalling isomorphism is equivalent to the previously studied linear relaxation of isomorphism known as *fractional isomorphism* [22].

Another approach we take is to develop characterizations of isomorphism and quantum isomorphism in terms of conic feasibility programs over the completely positive and completely positive semidefinite cones respectively. This is similar to work done in [13, 24, 25], but in our case the programs can be somewhat simplified due the highly structured form of the isomorphism game. By relaxing to either the doubly nonnegative ($\mathcal{DNN}$) or positive semidefinite ($\mathcal{S}_+$) cones, we are able to use this conic feasibility program to define $\mathcal{DNN}$- and $\mathcal{S}_+$-isomorphism, denoted $\cong_{\mathcal{DNN}}$ and $\cong_{\mathcal{S}_+}$ respectively. Interestingly, these semidefinite relaxations of quantum isomorphism are still stronger than non-signalling isomorphism. Therefore, for any graphs G and H we have that $G \cong H \Rightarrow G \cong_q H \Rightarrow G \cong_{\mathcal{DNN}} H \Rightarrow G \cong_{\mathcal{S}_+} H \Rightarrow G \cong_{ns} H$. Moreover, we are able to show that none of these implications can be reversed. In particular, we give a general method for constructing quantum isomorphic graphs that are not isomorphic, based on binary linear systems that are not satisfiable but are *quantum satisfiable*.

Interestingly, the notion of $\mathcal{DNN}$-isomorphism turns out to be equivalent to a previously studied relation in graph theory. This relation, known as graph *equivalence*, is defined in terms of an isomorphism between a certain matrix algebra associated to each of the graphs. To prove this equivalence, the main idea is to use the matrix in the conic feasibility program definition of $\mathcal{DNN}$-isomorphism as the Choi matrix of a linear map from the space of matrices indexed by $V(G)$ to the space of matrices indexed by $V(H)$. Like fractional isomorphism, there exists a polynomial time algorithm for deciding graph equivalence.

We also prove a similar algebraic characterization for $\mathcal{S}_+$-isomorphism, but this appears to be a new relation. However, we are able to prove that this relation is strictly stronger than cospectrality. We also show that classical, quantum, $\mathcal{DNN}$-, and $\mathcal{S}_+$-isomorphism can be characterized in terms of whether the Lovász theta function, or an appropriate generalization, achieves a particular value on a certain product graph. This is similar to the results of [24].

The full version of this work, including all of the proofs, is given in [3] and [16].

2 The Graph Isomorphism Game

Given graphs G and H, the (G, H)-isomorphism game is a nonlocal game whose inputs and outputs are vertices of the graphs G and H. For a detailed explanation of general nonlocal games see the full version of the paper [3]. The (G, H)-isomorphism game is played as follows: A referee/verifier selects uniformly at random a pair of vertices $x_A, x_B \in V(G) \cup V(H)$ and sends x_A to Alice and x_B to Bob respectively. The players respond with vertices $y_A, y_B \in V(G) \cup V(H)$. Throughout, we assume that $V(G)$ and $V(H)$ are disjoint so that players know which graph their vertex is from. As with any nonlocal game, Alice and Bob may agree on a strategy for playing the game beforehand, but they are not allowed to communicate after the game has commenced. In order to concisely state the conditions under which Alice and Bob win the (G, H)-isomorphism game, we let $\mathrm{rel}(g, g')$, for vertices g, g' of some graph G, denote the *relationship* of the vertices g and g', i.e., whether they are equal, adjacent, or distinct and non-adjacent.

The first winning condition is that each player must respond with a vertex from the graph that the vertex they received was *not* from. In other words we require that:

$$x_A \in V(G) \Leftrightarrow y_A \in V(H) \text{ and } x_B \in V(G) \Leftrightarrow y_B \in V(H)\,. \tag{1}$$

If condition (1) is not met, the players lose. Assuming (1) holds we define g_A to be the unique vertex of G among x_A and y_A, and we define g_B, h_A, and h_B similarly. In order to win, the answers of the players must also satisfy the following condition:

$$\mathrm{rel}(g_A, g_B) = \mathrm{rel}(h_A, h_B)\,. \tag{2}$$

In other words, if Alice and Bob are given the same vertex, then they must respond with the same vertex. If they receive (non-)adjacent vertices, they must return (non-)adjacent vertices. Also, assuming that Alice receives g_A and Bob h_B, Alice's output h_A must be related to h_B the same way Bob's output g_B is related to g_A. Note that we do not explicitly require that G and H have the same number of vertices. It is also worth pointing out that the (G, H)-isomorphism game is equivalent to the $(\overline{G}, \overline{H})$-isomorphism game, where $\overline{G}$ denotes the *complement* of G, i.e., the graph obtained by switching edges and non-edges of G.

In general one may be interested in the best probability with which Alice and Bob can win this game for some particular G and H. In this work however, we will only be interested in whether or not they can win *perfectly*, i.e., with probability 1. Thus, from henceforth when we say that Alice and Bob cam win the (G, H)-isomorphism game, we mean that they can win with probability 1. Similarly, a *winning* or *perfect strategy* is one that allows them to win with certainty.

Given any fixed strategy for the (G, H)-isomorphism game, we denote by $p(y_A, y_B|x_A, x_B)$ the joint conditional probability of Alice and Bob responding with y_A and y_B upon receiving inputs x_A and x_B respectively. We call such a joint conditional probability distribution a *correlation*. An easy but important observation is that a given strategy for the (G, H)-

isomorphism game is perfect if and only if its corresponding correlation p satisfies

$$p(y_A, y_B|x_A, x_B) = 0, \text{ whenever conditions (1) or (2) fail.} \tag{3}$$

In this work we will focus on three classes of strategies/correlations for the isomorphism game: classical, quantum, and non-signalling.

2.1 Classical strategies

A *deterministic classical strategy* for a nonlocal game is one in which Alice's response is determined by her input, and similarly for Bob. In a general classical strategy, the players may use shared randomness to determine their responses. Once the value that their shared randomness takes is fixed, Alice and Bob's strategy becomes deterministic. This means that the set of (perfect) classical correlations is equal to the convex hull of (perfect) classical deterministic correlations.

Suppose that $\varphi : V(G) \to V(H)$ is an isomorphism of graphs G and H. Then we can construct a perfect strategy for the (G, H)-isomorphism game as follows: if Alice receives $g \in V(G)$ as her input, then she responds with $\varphi(g)$ as her output, and if she receives $h \in V(H)$ as her input, then she responds with $\varphi^{-1}(h)$ as her output. Bob behaves identically. It is not hard to see that this allows Alice and Bob to win the game perfectly. It is not much more difficult to prove the converse (see [3]), and thus we have the following:

▶ **Theorem 1.** *For graphs G and H, the (G, H)-isomorphism game can be won perfectly with a classical strategy if and only if $G \cong H$.*

3 Quantum Isomorphism

In a quantum strategy for the (G, H)-isomorphism game, Alice and Bob are allowed to share and make joint measurements on an entangled state (see [3] for a detailed explanation of shared states and measurements). As any classical post-processing of the outcomes that Alice and Bob perform can be incorporated into the measurements themselves, we may assume that both Alice and Bob have a measurement for each input (an element of $V(G) \cup V(H)$) whose outcomes are indexed by their possible outputs (elements of $V(G) \cup V(H)$). Upon receiving input x, Alice performs her measurement corresponding to x and obtains some outcome y, which she uses as her output, and Bob behaves similarly. Formally, for each $x \in V(G) \cup V(H)$, Alice has a measurement $\mathcal{E}_x = \{E_{xy} \in \mathbb{C}^{d_A \times d_A} : y \in V(G) \cup V(H)\}$ where $E_{xy} \succeq 0$ and $\sum_{y \in V(G) \cup V(H)} E_{xy} = I$, and similarly Bob has measurement $\mathcal{F}_x = \{F_{xy} \in \mathbb{C}^{d_B \times d_B} : y \in V(G) \cup V(H)\}$. They perform these measurements on their shared state $\psi \in \mathbb{C}^{d_A} \otimes \mathbb{C}^{d_B}$. Note that there are no restrictions on $d_A, d_B \in \mathbb{N}$. The corresponding correlation p for this strategy is given by $p(y, y'|x, x') = \psi^\dagger \left(E_{xy} \otimes F_{x'y'}\right) \psi$. If there exists such a strategy that allows Alice and Bob to win the (G, H)-isomorphism game perfectly, then we say that G and H are *quantum isomorphic* and write $G \cong_q H$.

An important property of the isomorphism game is that if Alice and Bob are given the same inputs, they must respond with the same outputs. Games with this property are called *synchronous*, and the perfect quantum strategies for such games are known to have a particular form [14, 23, 15, 8, 21]. Using this we are able to give the following reformulation of quantum isomorphism, the full proof of which is given in [3]:

▶ **Theorem 2.** *Let G and H be graphs. Then $G \cong_q H$ if and only if there exists $d \in \mathbb{N}$ and orthogonal projectors $E_{gh} \in \mathbb{C}^{d \times d}$ for $g \in V(G)$ and $h \in V(H)$ such that*

(i) $\sum_{h\in V(H)} E_{gh} = I$, *for all* $g \in V(G)$;
(ii) $\sum_{g\in V(G)} E_{gh} = I$, *for all* $h \in V(H)$;
(iii) $E_{gh}E_{g'h'} = 0$, *if* $\mathrm{rel}(g,g') \neq \mathrm{rel}(h,h')$.

The projectors in the above theorem correspond to Alice's (or Bob's) measurement operators in a perfect quantum strategy. One consequence of this is that any winning quantum correlation p for the (G,H)-isomorphism game is *input-output symmetric*, i.e.,

$$p(y,y'|x,x') = p(x,y'|y,x') = p(y,x'|x,y') = p(x,x'|y,y') \text{ for all } x,x',y,y' \in V(G)\cup V(H).$$

Given a set of projectors as in the theorem above, one thing that we could do with them is to make them elements of a block matrix with rows indexed by $V(G)$ and columns indexed by $V(H)$. Investigating the properties of such a matrix leads to the following definition:

▶ **Definition 3.** A block matrix $\mathcal{P}$ with blocks of size d is called a *projective permutation matrix (of block size d)* if it is unitary and all of its blocks are orthogonal projectors.

Note that a projective permutation matrix of block size one is a unitary matrix whose entries square to themselves, *i.e.*, a permutation matrix. The following lemma (see [3] for full proof) shows that projective permutation matrices can be built out of projectors satisfying the first two conditions of Theorem 2.

▶ **Lemma 4.** *A block matrix* $\mathcal{P}$ *with blocks* E_{ij} *for* $i,j \in [n]$ *is a projective permutation matrix if and only if the matrix* E_{ij} *is a projector for all* $i,j \in [n]$ *and*
(i) $\sum_{j=1}^{n} E_{ij} = I$, *for all* $i \in [n]$;
(ii) $\sum_{i=1}^{n} E_{ij} = I$, *for all* $j \in [n]$.

We note that in the special case where all of the blocks E_{ij} of a projective permutation matrix are of rank one, then the unit vectors that the E_{ij} project onto form a *quantum Latin square*, a notion introduced in [18].

A useful reformulation of graph isomorphism can be stated in terms of the adjacency matrices of the corresponding graphs. Given a graph G, the adjacency matrix of G, denoted A_G, is the symmetric 01-matrix whose gg'-entry is 1 if and only if g is adjacent to g', which we denote by $g \sim g'$. Graphs G and H are isomorphic if and only if there exists a permutation matrix P such that $A_G P = PA_H$, or equivalently $P^T A_G P = A_H$. The motivation for considering projective permutation matrices is that they play the role of permutation matrices in an analogous formulation for quantum isomorphism. This is made precise by the following theorem, whose proof is given in [3]:

▶ **Theorem 5.** *For any two graphs* G *and* H *we have that* $G \cong_q H$ *if and only if there exists* $d \in \mathbb{N}$ *and a projective permutation matrix* $\mathcal{P}$ *of block size* d *such that*

$$(A_G \otimes I_d)\mathcal{P} = \mathcal{P}(A_H \otimes I_d). \tag{4}$$

Since projective permutation matrices are unitary, we can rewrite Equation (4) as $\mathcal{P}^\dagger(A_G \otimes I_d)\mathcal{P} = (A_H \otimes I_d)$. Again, since $\mathcal{P}$ is unitary, this implies that $A_G \otimes I_d$ and $A_H \otimes I_d$, and thus also A_G and A_H, have the same multiset of eigenvalues. Thus we have the following:

▶ **Corollary 6.** *If* $G \cong_q H$, *then* G *and* H *are cospectral with cospectral complements.*

3.1 Separating Classical and Quantum Isomorphism

In order to construct graphs that are quantum isomorphic but not isomorphic, we introduce a type of game investigated by Cleve and Mittal [9] known as *binary constraint system* (BCS) games. We will show that, in the linear case, one can reduce the existence of a perfect classical (quantum) strategy for a BCS game to the existence of a perfect classical (quantum) strategy to a corresponding isomorphism game.

A linear binary constraint system (BCS) $\mathcal{F}$ consists of a family of binary variables $x_1, \ldots, x_n$ and *constraints* $C_1, \ldots, C_m$, where each C_ℓ is a linear equation over $\mathbb{F}_2$ in some subset of the variables. Thus C_ℓ takes the form $\sum_{x_i \in S_\ell} x_i = b_\ell$ for some $S_\ell \subseteq \{x_1, \ldots, x_n\}$ and $b_\ell \in \{0, 1\}$. We say that a BCS is *satisfiable* if there is an assignment of values from $\mathbb{F}_2$ to the variables x_i such that every constraint C_ℓ is satisfied. Such an assignment is known as a *satisfying assignment*.

An example of a linear BCS is the following:

$$\begin{aligned} x_1 + x_2 + x_3 &= 0 & x_1 + x_4 + x_7 &= 0 \\ x_4 + x_5 + x_6 &= 0 & x_2 + x_5 + x_8 &= 0 \\ x_7 + x_8 + x_9 &= 0 & x_3 + x_6 + x_9 &= 1 \end{aligned} \tag{5}$$

where addition is over $\mathbb{F}_2$. Note that the BCS given above is not satisfiable. Indeed, every variable appears in exactly two constraints and thus summing up all equations modulo 2 we get $0 = 1$.

To any linear BCS $\mathcal{F}$ we associate the following nonlocal game, which we call the *BCS game*. In the BCS game, the verifier gives Alice a constraint C_ℓ and Bob a constraint C_k. In order to win, they must each respond with an assignment of values to the variables in their respective constraints such that those constraints are satisfied. Furthermore, for the variables in $S_\ell \cap S_k$, Alice and Bob must agree on their assignment. Note that if they are given the same constraint, these conditions imply that they must give the same response. We note that in [9], Cleve and Mittal also define a nonlocal game for any linear BCS. This game is very similar, though not identical to the above (Bob is only asked single variables in the game of [9]). However, their results imply that in the quantum and classical cases, these two games are equivalent.

As with the other nonlocal games we have considered in this work, it is not difficult to see that Alice and Bob can win the BCS game classically with probability 1 if and only if the corresponding BCS is satisfiable. This motivates the following definition.

▶ **Definition 7.** A linear BCS is called *quantum satisfiable* if there exists a perfect quantum strategy for the corresponding BCS game.

To any linear BCS $\mathcal{F}$ with m constraints we associate the graph $G_\mathcal{F}$ which is defined as follows: For each constraint C_ℓ, and each assignment $f : S_\ell \to \mathbb{F}_2$ that *satisfies* C_ℓ we include a vertex (ℓ, f). Furthermore, we add an edge between two vertices (ℓ, f) and (k, f') if they are *inconsistent*, *i.e.*, if there exists $x_i \in S_\ell \cap S_k$ such that $f(x_i) \neq f'(x_i)$. We remark that this construction is related to the FGLSS reduction from [10], which is well known in approximability literature.

Given any linear BCS $\mathcal{F}$, we define the *homogenization* of $\mathcal{F}$, denoted by $\mathcal{F}_0$, to be the linear BCS obtained from $\mathcal{F}$ by changing the righthand sides of all of the constraints to 0. Note that the homogenization of a linear BCS always has a solution, namely the all-zero assignment. Also note that $G_\mathcal{F}$ and $G_{\mathcal{F}_0}$ have the same number of vertices.

Using these constructions, we are able to prove the following (see [3] for proof), where $\alpha(G)$ denotes the independence number of the graph G:

▶ **Theorem 8.** *Let $\mathcal{F}$ be a linear BCS with m constraints. Then the following are equivalent:*

(i) *$\mathcal{F}$ is satisfiable;*
(ii) *The graphs $G_{\mathcal{F}}$ and $G_{\mathcal{F}_0}$ are isomorphic;*
(iii) *$\alpha(G_{\mathcal{F}}) = m$.*

Using the notions of quantum independence number, denoted α_q, and projective packings, we can also prove the following quantum analog of the above (see [3]):

▶ **Theorem 9.** *Let $\mathcal{F}$ be a linear BCS with m constraints. Then the following are equivalent:*

(i) *$\mathcal{F}$ is quantum satisfiable;*
(ii) *The graphs $G_{\mathcal{F}}$ and $G_{\mathcal{F}_0}$ are quantum isomorphic;*
(iii) *There exists a projective packing of $G_{\mathcal{F}}$ of value m;*
(iv) *$\alpha_q(G_{\mathcal{F}}) = m$.*

Thus, to find a pair of graphs that are quantum isomorphic but not isomorphic, it suffices to find a linear BCS that is quantum satisfiable but not satisfiable. One such example is the one given in (5), which corresponds to the well-known Mermin-Peres magic square game. The pair of graphs obtained from this BCS are shown in [3]. In fact, Arkhipov has shown how to construct such a BCS from any non-planar graph [1].

We note here that the first separating example, which was found with the help of Albert Atserias, was slightly different than the one presented above. It was a version of the celebrated CFI construction, named after Cai, Fürer and Immerman [7]. The original CFI construction was designed to produce pairs of non-isomorphic graphs that cannot be distinguished by the d-dimensional Weisfeiler-Lehman algorithm for any fixed d. The CFI construction was reinterpreted by Atserias, Bulatov, and Dawar [2] to view it as an encoding of special systems of linear equations over $\mathbb{Z}_2$, where each variable appears in precisely two equations. Our first separating example was literally the CFI construction corresponding to a system of linear equations as in [2], in which each variable appears in exactly two equations, and that is classically unsatisfiable over $\mathbb{Z}_2$ but quantum satisfiable. The Mermin-Peres magic square game gives rise to such a system of linear equations. The final construction which we described above is a simplified version of this, in which several vertices have been merged together, and several others have been removed, without changing the outcome. The final graphs have a few dozens of vertices. As it turns out, this streamlined version of the construction is quite similar to the FGLSS reduction from the theory of hardness of approximation [10], which interpreted in this context is a reduction from the feasibility problem for arbitrary systems of linear equations over $\mathbb{Z}_2$ to the graph isomorphism problem. As it turns out, the FGLSS construction was also used in the context of the graph isomorphism problem in [19].

4 Non-signalling Isomorphism

An important property of any quantum strategy for the (G, H)-isomorphism game (or any nonlocal game), is that it does not allow the players to communicate any information about their inputs to one another. Formally, this corresponds to

$$\begin{aligned}
&\sum_{y_B} p(y_A, y_B|x_A, x_B) = \sum_{y_B} p(y_A, y_B|x_A, x'_B), \text{ for all } x_A, y_A, x_B, x'_B, \text{ and} \\
&\sum_{y_A} p(y_A, y_B|x_A, x_B) = \sum_{y_A} p(y_A, y_B|x'_A, x_B), \text{ for all } x_B, y_B, x_A, x'_A
\end{aligned} \tag{6}$$

Any correlation which obeys this condition is known as *non-signalling*, and this is known to be a strictly larger class than quantum correlations. Note that this is a condition on correlations rather than strategies, and indeed there may not be any way to physically realize a given non-signalling correlation. Still, there are good reasons for considering this class of correlations. First, they are a linear relaxation of quantum correlations, and so they often allow us to obtain useful bounds on what is possible with quantum strategies, which are notoriously difficult to analyze. Second, they are interesting in their own right since they represent the extreme class of correlations in two senses. In the physical sense, the non-signalling condition can be thought of as encoding the notion that nothing, including information, can travel faster than the speed of light. Thus if Alice and Bob are separated by a great distance and must respond with their answers within a short window of time, then their strategy must be non-signalling. From a mathematical perspective, non-signalling correlations are the most general class of correlations it makes sense to consider for nonlocal games since any larger class would, by definition, allow the parties to communicate to a certain extent. This would essentially violate the definition of a nonlocal game which requires that the parties cannot communicate.

Using the non-signalling condition and the winning conditions of the isomorphism game, one can prove the following lemma (proof given in [3]):

▶ **Lemma 10.** *Let p be a winning non-signalling correlation for the (G, H)-isomorphism game. Then $p(h, h|g, g) = p(g, h|h, g) = p(h, g|g, h) = p(g, g|h, h)$, for all $g \in V(G)$, $h \in V(H)$.*

Note that for a winning correlation p for the (G, H)-isomorphism game, for $g \in V(G)$ we have that $p(y, y'|g, x') = 0$ unless $y \in V(H)$, and similarly with Alice and Bob or G and H switched. This, along with the above lemma allows us to take any winning non-signalling correlation for the (G, H)-isomorphism game and construct the following *doubly stochastic* matrix: $D_{gh} = p(h, h|g, g)$.

It turns out that this matrix has the interesting property that $A_G D = D A_H$. Whenever such a doubly stochastic matrix exists, one says that G and H are *fractionally isomorphic*, denoted $G \cong_f H$. Thus, non-signalling isomorphic graphs are always fractionally isomorphic.

To prove the converse of the above, we need a result of Ramana, Scheinerman, and Ullman [22] which shows that fractional graph isomorphism is equivalent to deciding whether the graphs have a common equitable partition. To explain this result we first need to introduce some definitions.

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a partition of $V(G)$ for some graph G. The partition $\mathcal{C}$ is called *equitable* if there exist numbers c_{ij} for $i, j \in [k]$ such that any vertex in C_i has exactly c_{ij} neighbors in C_j. Note that c_{ij} and c_{ji} are not necessarily equal, but $c_{ij}|C_i| = c_{ji}|C_j|$. We refer to the numbers c_{ij} as the *partition numbers* of an equitable partition $\mathcal{C}$. A trivial example of this is the partition where each part has size 1. Less trivially, if G is regular, the partition with only one cell is equitable.

Equivalently, a partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ is equitable if for any $i \in [k]$, the subgraph induced by the vertices in C_i is regular, and for any $i \neq j \in [k]$ the subgraph with vertex set $C_i \cup C_j$ and containing the edges between C_i and C_j is a semiregular bipartite graph.

We say that $\mathcal{C}$ and $\mathcal{D}$ have a *common equitable partition* if there exist equitable partitions $\mathcal{C} = \{C_1, \ldots, C_k\}$ and $\mathcal{D} = \{D_1, \ldots, D_{k'}\}$ for G and H respectively, satisfying $k = k'$, $|C_i| = |D_i|$ for all $i \in [k]$, and lastly, $c_{ij} = d_{ij}$ for all $i, j \in [k]$. As an example, if G and H are both d-regular and have the same number of vertices, then the single cell partitions form a common equitable partition, and thus, by Theorem 11, any such graphs are fractionally isomorphic. This makes it seem like fractional isomorphism is a weak condition, but in fact it

is known [6] that asymptotically almost surely no graphs are fractionally isomorphic to any graphs that they are not isomorphic to. Since non-signalling/fractional isomorphism is the coarsest relation we will consider in this work, the same holds for all the other relations we will see. As mentioned above, common equitable partitions characterize fractional isomorphism.

▶ **Theorem 11** ([22]). *Two graphs are fractionally isomorphic if and only if they have a common equitable partition.*

Given a common equitable partition $\mathcal{C} = \{C_1, \dots, C_k\}$ and $\mathcal{D} = \{D_1, \dots, D_k\}$ for graphs G and H respectively, one can construct a perfect non-signalling strategy for the (G, H)-isomorphism game. The details are given in [3], but the idea is that if Alice is given $g \in C_i$ and Bob given $g' \in C_j$ and g and g' are adjacent/non-adjacent/equal, then they respond uniformly at random with $h \in D_i$ and $h' \in D_j$ that are adjacent/non-adjacent/equal. The fact that the corresponding correlation is non-signalling follows from the fact that $\mathcal{C}$ and $\mathcal{D}$ form a common equitable partition of G and H. Therefore, we have the following:

▶ **Theorem 12.** *For any graphs G and H we have that $G \cong_f H$ if and only if $G \cong_{ns} H$.*

5 Conic Formulations

Given graphs G and H and a winning correlation p for the (G, H)-isomorphism game, define the matrix M^p to be the matrix with rows and columns indexed by $V(G) \times V(H)$ to have entries $M^p_{gh,g'h'} = p(h, h'|g, g')$.

Note that the matrix M^p does not contain all of the probabilities of p, only those corresponding to inputs from $V(G)$ and outputs from $V(H)$. Thus, in general the matrix M^p may not completely determine the correlation p. However, if p is input-output symmetric, as in the case of classical or quantum correlations, then p is determined by the matrix M^p. Also note that in the classical and quantum cases Alice and Bob are symmetric, i.e., $p(y, y'|x, x') = p(y', y|x', x)$ for all $x, x', y, y' \in V(G) \cup V(H)$, and thus M^p is symmetric.

Since p is a correlation, sums of certain entries of M^p must be 1. Furthermore, since p is winning, certain entries of M^p must be 0. This motivates the following definition:

▶ **Definition 13.** Let G and H be graphs and $\mathcal{K}$ a matrix cone. We say that a matrix M with rows and columns indexed by $V(G) \times V(H)$ is a *$\mathcal{K}$-isomorphism matrix for G to H* if $M \in \mathcal{K}$ and

$$\sum_{h,h' \in V(H)} M_{gh,g'h'} = 1 \text{ for all } g, g' \in V(G) \tag{7}$$

$$\sum_{g,g' \in V(G)} M_{gh,g'h'} = 1 \text{ for all } h, h' \in V(H) \tag{8}$$

$$M_{gh,g'h'} = 0 \text{ if } \mathrm{rel}(g, g') \neq \mathrm{rel}(h, h'). \tag{9}$$

We will say that graphs G and H are $\mathcal{K}$-isomorphic, and write $G \cong_{\mathcal{K}} H$, whenever there exists a $\mathcal{K}$-isomorphism matrix for G to H.

Though we have defined them for any matrix cone $\mathcal{K}$, we will mainly be interested in just four cones in this work. The first cone is the positive semidefinite cone, denoted $\mathcal{S}_+$. Recall that a matrix M is positive semidefinite if and only if it is the Gram matrix of a set of vectors $v_1, \dots, v_n$, i.e., $M_{ij} = v_i^T v_j$. We will also be interested in the *doubly nonnegative cone*, denoted $\mathcal{DNN}$, which consists of all positive semidefinite matrices that are also entrywise nonnegative. The next cone we will consider is the recently introduced [13] *completely positive semidefinite*

cone, denoted $\mathcal{CS}_+$, which will correspond to quantum correlations. A matrix M is completely positive semidefinite if it is the Gram matrix of positive semidefinite matrices $\rho_1, \ldots, \rho_n$, i.e., $M_{ij} = \langle \rho_i, \rho_j \rangle := \mathrm{Tr}(\rho_i^\dagger \rho_j)$. Note that this inner product is equal to the usual inner product if we think of the matrices ρ_i as vectors, and thus $\mathcal{CS}_+ \subseteq \mathcal{S}_+$. Moreover, $\mathrm{Tr}(AB) \geq 0$ for any positive semidefinite matrices A and B, with equality if and only if $AB = 0$, and thus $\mathcal{CS}_+ \subseteq \mathcal{DNN}$. Lastly, the *completely positive cone*, denoted $\mathcal{CP}$, will correspond to the classical correlations. A matrix is completely positive if it is the Gram matrix of entrywise nonnegative vectors $v_1, \ldots, v_n$. Note that if D_i is the diagonal matrix with diagonal entries equal to the entries of v_i, then D_i is positive semidefinite and $v_i^T v_j = \mathrm{Tr}(D_i D_j)$. Therefore, $\mathcal{CP} \subseteq \mathcal{CS}_+$. Altogether we have that $\mathcal{CP} \subseteq \mathcal{CS}_+ \subseteq \mathcal{DNN} \subseteq \mathcal{S}_+$. Moreover, it is known that these containments are all strict for square matrices of dimension at least 5. With the above notions, we can prove the following (full proof given in [16]):

▶ **Theorem 14.** *Suppose G and H are graphs and p is a correlation for the (G, H)-isomorphism game. Then p is winning classical correlation if and only if p is input-output symmetric and M^p is a $\mathcal{CP}$-isomorphism matrix.*

▶ **Theorem 15.** *Suppose G and H are graphs and p is a correlation for the (G, H)-isomorphism game. Then p is a winning quantum correlation if and only if p is input-output symmetric and M^p is a $\mathcal{CS}_+$-isomorphism matrix.*

The above motivates us to investigate the notions of $\mathcal{DNN}$- and $\mathcal{S}_+$-isomorphism. In order to do this, we need to define something we call an isomorphism map.

After submission of this work, a referee informed us that the graph isomorphism problem has been formulated as a completely positive program previously in [11].

5.1 Isomorphism Maps

Given a $\mathcal{K}$-isomorphism matrix M for G to H, the *isomorphism map* Φ_M is a linear map from the space of complex matrices indexed by $V(G)$ to the space of complex matrices indexed by $V(H)$ defined as $(\Phi_M(X))_{h,h'} = \sum_{g,g'} M_{gh,g'h'} X_{g,g'}$.

For $\mathcal{K} \subseteq \mathcal{S}_+$, this map has some remarkable properties. In particular, it is completely positive, meaning that $\Phi_M \otimes \mathrm{id}$ maps psd matrices to psd matrices, where id can be an identity map of any size. This is not related to the completely positive cone, an unfortunate ambiguity. The map Φ_M is trace-preserving and unital, meaning that $\Phi_M(I) = I$. It also preserves the sum of the entries of a matrix, and maps the all ones matrix J to itself. If $\mathcal{K} \in \mathcal{DNN}$, then Φ_M maps entrywise nonnegative matrices to entrywise nonnegative matrices. These last three properties define a notion of being doubly stochastic purely in terms of linear maps. The adjoint of an isomorphism map from G to H is an isomorphism map from H to G. Lastly, one can show that $\Phi_M(A_G) = A_H$ and $\Phi_M^*(A_H) = A_G$, where Φ_M^* is the adjoint of Φ_M which will be an isomorphism map for H to G. Since the eigenvalues of a Hermitian matrix X majorize those of a Hermitian matrix Y if and only if there exists a completely positive, trace-preserving, unital map taking X to Y, this last property implies Lemma 16 below. None of these properties are difficult to show, and the details are given in [16].

▶ **Lemma 16.** *If G and H are $\mathcal{S}_+$-isomorphic graphs, then they are cospectral.*

The idea of isomorphism maps is borrowed from Ortiz and Paulsen who constructed similar linear maps from winning correlations for the *homomorphism* game in [20]. These isomorphism maps will allow us to give characterizations of $\mathcal{DNN}$- and $\mathcal{S}_+$-isomorphisms in terms of certain algebras associated to graphs. But first we must introduce these algebras.

5.2 Coherent and Partially Coherent Algebras

A subspace of $\mathbb{C}^{n\times n}$ which is also closed under matrix multiplication is an *algebra*. If $\mathcal{A}$ is a subalgebra of $\mathbb{C}^{n\times n}$, then $\mathcal{A}$ is a *coherent algebra* if it contains the identity and the all ones matrix, is closed under Schur (entrywise) product, and is closed under conjugate transpose, *i.e.*, is self-adjoint. The simplest example of a coherent algebra is $\text{span}\{I, J-I\}$. Of course, $\mathbb{C}^{n\times n}$ is itself a coherent algebra. Less trivially, if A is the adjacency matrix of any *strongly regular graph*, then $\text{span}\{I, A, J-I-A\}$ is a coherent algebra.

It follows from the fact that a coherent algebra $\mathcal{A}$ is closed under Schur product that it must have an orthogonal (with respect to the Hilbert-Schmidt inner product) basis of 01 matrices $A_1, \ldots, A_r$. To each of the matrices A_i, we can associate a subset of $V(G)\times V(G)$, namely the set of ordered pairs (g, g') such that the gg'-entry of A_i is 1. This gives a partition of the ordered pairs of vertices of G. One can reformulate the properties of being a coherent algebra in terms of this partition, and a partition with these properties is known as a *coherent configuration*. Conversely, any coherent configuration corresponds to some coherent algebra. The parts in a coherent configuration are usually referred to as its *classes*.

Coherent algebras of graphs

It is not hard to see that the intersection of two coherent algebras is a coherent algebra. We can therefore define the *coherent algebra of a graph* G, denoted $\mathcal{A}_G$, to be the intersection of all coherent algebras containing its adjacency matrix A_G, *i.e.*, the smallest coherent algebra containing A_G. Equivalently, this is the set of all matrices that can be written as a finite expression involving I, A, J, and the operations of addition, scalar multiplication, matrix multiplication, Schur multiplication, and conjugate transpose.

An *isomorphism* between coherent algebras $\mathcal{A}$ and $\mathcal{B}$ is a bijective linear map $\phi : \mathcal{A} \to \mathcal{B}$ that preserves all operations of a coherent algebra, *i.e.*,

- $\phi(M^\dagger) = \phi(M)^\dagger$ for all $M \in \mathcal{A}$;
- $\phi(MN) = \phi(M)\phi(N)$ for all $M, N \in \mathcal{A}$;
- $\phi(M \bullet N) = \phi(M) \bullet \phi(N)$ for all $M, N \in \mathcal{A}$.

As a consequence of the above, we must have that $\phi(I) = I$ and $\phi(J) = J$. More generally, if ϕ is an isomorphism of coherent algebras $\mathcal{A}$ and $\mathcal{B}$, then ϕ maps the elements of the unique 01 basis of $\mathcal{A}$ to those of $\mathcal{B}$ in a manner that preserves how the basis elements relate to one another (this is made precise in [16]).

▶ **Definition 17.** If G and H are two graphs with respective adjacency matrices A_G and A_H and coherent algebras $\mathcal{A}_G$ and $\mathcal{A}_H$, then we say that G and H are *equivalent* if there exists an isomorphism ϕ from $\mathcal{A}_G$ to $\mathcal{A}_H$ such that $\phi(A_G) = A_H$. We refer to the map ϕ as an *equivalence* of G and H.

Note that the condition $\phi(A_G) = A_H$ completely determines the function ϕ on $\mathcal{A}_G$. In Section 5.4, we show that two graphs are $\mathcal{DNN}$-isomorphic if and only if they are equivalent.

5.3 Partially Coherent Algebras

Suppose that S is some subset of $\mathbb{C}^{n\times n}$. We say that an algebra $\mathcal{A}$ is an *S-partially coherent algebra* if $\mathcal{A}$ contains the identity, is self-adjoint, contains the all ones matrix, and is closed under Schur multiplication by any matrix in S.

As with coherent algebras, it is easy to see that the intersection of two S-partially coherent algebras is an S-partially coherent algebra. Therefore, there is some minimal S-partially

coherent algebra for any S. This will be equal to the set of matrices that can be expressed using the elements of $S \cup \{I, J\}$ and a finite number of the operations of addition, scalar multiplication, matrix multiplication, conjugate transposition, and Schur multiplication where at least one of the factors is an element of S.

We define the *partially coherent algebra of a graph* G, denoted $\hat{\mathcal{A}}_G$, to be the minimal S-partially coherent algebra where $S = \{I, A_G\}$. Note that this will also be S'-partially coherent for $S' = \{I, A_G, A_{\overline{G}}\}$ since $A_{\overline{G}} = J - I - A_G$ and J is the Schur identity.

▶ **Definition 18.** Let G and H be graphs with adjacency matrices A_G and A_H and partially coherent algebras $\hat{\mathcal{A}}_G$ and $\hat{\mathcal{A}}_H$ respectively. We say that G and H are *partially equivalent* if there exists a linear bijection $\phi : \hat{\mathcal{A}}_G \to \hat{\mathcal{A}}_H$ such that

1. $\phi(M^\dagger) = \phi(M)^\dagger$ for all $M \in \hat{\mathcal{A}}_G$;
2. $\phi(MN) = \phi(M)\phi(N)$ for all $M, N \in \hat{\mathcal{A}}_G$;
3. $\phi(I) = I$, $\phi(A_G) = A_H$, and $\phi(J) = J$;
4. $\phi(M \bullet N) = \phi(M) \bullet \phi(N)$ for all $M \in \{I, A_G\}$ and $N \in \hat{\mathcal{A}}_G$.

We refer to ϕ as a *partial equivalence* of G and H.

5.4 Characterizations of $\mathcal{DNN}$- and $\mathcal{S}_+$-Isomorphisms

Using the ideas from the previous sections we can now give our characterizations of $\mathcal{DNN}$- and $\mathcal{S}_+$-isomorphisms (full proof given in [16]):

▶ **Theorem 19.** *Let G and H be graphs. Then $G \cong_{\mathcal{DNN}} H$ if and only if G and H are equivalent. Also, $G \cong_{\mathcal{S}_+} H$ if and only if G and H are partially equivalent.*

The proof of the above goes roughly as follows: If $G \cong_{\mathcal{DNN}} H$, then there exists a $\mathcal{DNN}$-isomorphism matrix M and corresponding isomorphism map Φ_M. When restricted to the coherent algebra $\mathcal{A}_G$, the map Φ_M is an equivalence of G and H. Conversely, suppose ϕ is an equivalence of G and H, and let Π be the orthogonal projection of $\mathbb{C}^{V(G)\times V(G)}$ to $\mathcal{A}_G$, then the Choi matrix of the map $\phi \circ \Pi$ is a $\mathcal{DNN}$-isomorphism matrix for G to H. The proof for $\mathcal{S}_+$-isomorphism is similar.

There is a well known algorithm, known as the Weisfeiler-Lehman algorithm, that determines whether two graphs are equivalent. Thus $\mathcal{DNN}$-isomorphism is polynomial time decidable. We do not yet know the complexity of $\mathcal{S}_+$-isomorphism, but we suspect it is also polynomial time decidable.

We can use the above characterizations of $\mathcal{DNN}$- and $\mathcal{S}_+$-isomorphisms to prove the following results for 1-walk-regular and distance regular graphs (proofs given in [16]):

▶ **Theorem 20.** *Let G be a connected 1-walk-regular graph. If H is a graph, then $G \cong_{\mathcal{S}_+} H$ if and only if H is a connected 1-walk-regular graph that is cospectral to G.*

▶ **Theorem 21.** *Let G be a distance regular graph. If H is a graph, then $G \cong_{\mathcal{DNN}} H$ if and only if H is a distance regular graph that is cospectral to G.*

▶ **Lemma 22.** *If $G \cong_{\mathcal{DNN}} H$, then G and H have the same radius and diameter.*

6 Separations

In Section 3.1 we saw that isomorphism and quantum isomorphism are distinct relations. In this section we show that the rest of relations we have defined are distinct from one another. Here we give examples and brief explanations, but the full details are in [16].

Quantum vs. $\mathcal{DNN}$-Isomorphism. The 4×4 rook's graph and the Shrikhande graph are cospectral distance regular graphs. Therefore, they are $\mathcal{DNN}$-isomorphic by Theorem 21. However, we show that their complements have different quantum chromatic numbers, a parameter that is preserved by quantum isomorphism (see [3] for details).

$\mathcal{DNN}$-Isomorphism vs. $\mathcal{S}_+$-Isomorphism. The 4-cube graph has the binary strings of length 4 as its vertices, two being adjacent if they differ in exactly one bit. The Hoffman graph is the unique cospectral mate of the 4-cube, and they are both connected and 1-walk-regular. Therefore they are $\mathcal{S}_+$-isomorphic by Theorem 20. However, the 4-cube has radius 4 and the Hoffman graph has radius 3, thus they are not $\mathcal{DNN}$-isomorphic by Lemma 22.

$\mathcal{S}_+$-Isomorphism vs. Non-signalling Isomorphism. By Lemma 16, any pair of k-regular graphs on n vertices for some n and k that are not cospectral will work for this. For example, the 6-cycle and two disjoint 3-cycles will do.

Acknowledgements. The authors would like to especially thank Albert Atserias who was instrumental in finding the first separation between isomorphism and quantum isomorphism.

References

1 Alex Arkhipov. Extending and characterizing quantum magic games, 2012. arXiv:1209.3819.

2 Albert Atserias, Andrei Bulatov, and Anuj Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666–1683, 2009. A preliminary version appeared in ICALP 2007. doi:10.1016/j.tcs.2008.12.049.

3 Albert Atserias, Laura Mančinska, David E. Roberson, Robert Šámal, Simone Severini, and Antonios Varvitsiotis. Quantum and non-signalling graph isomorphisms, 2016. URL: https://arxiv.org/abs/1611.09837, arXiv:1611.09837.

4 László Babai. Automorphism groups, isomorphism, reconstruction. In *Handbook of Combinatorics (Vol. 2)*, pages 1447–1540. MIT Press, 1995. doi:233228.233236.

5 László Babai. Graph isomorphism in quasipolynomial time, 2015. arXiv:1512.03547.

6 László Babai, Paul Erdős, and Stanley M. Selkow. Random graph isomorphism. *SIAM Journal on Computing*, 9(3):628–635, 1980. doi:10.1137/0209047.

7 Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. doi:10.1007/BF01305232.

8 Peter J. Cameron, Ashley Montanaro, Michael W. Newman, Simone Severini, and Andreas Winter. On the quantum chromatic number of a graph. *Electronic Journal of Combinatorics*, 14(1), 2007. arXiv:quant-ph/0608016.

9 Richard Cleve and Rajat Mittal. Characterization of binary constraint system games. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, ICALP'14, pages 320–331. Springer, 2014. arXiv:1209.2729.

10 Uriel Feige, Shafi Goldwasser, László Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS'91, pages 2–12, 1991. doi:10.1109/SFCS.1991.185341.

11 Luuk Gijben. *On approximations, complexity, and applications for copositive programming.* PhD thesis, 2015.

12 Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory.* Draft manuscript, 2013. Available online.

13 Monique Laurent and Teresa Piovesan. Conic approach to quantum graph parameters using linear optimization over the completely positive semidefinite cone. *SIAM Journal on Optimization*, 25(4):2461–2493, 2015. doi:10.1137/14097865X.

14 Laura Mančinska and David E. Roberson. Quantum homomorphisms. *Journal of Combinatorial Theory, Series B*, 118:228–267, 2016. arXiv:1212.1724.

15 Laura Mančinska, David E. Roberson, and Antonios Varvisotis. On deciding the existence of perfect entangled strategies for nonlocal games. *Chicago Journal of Theoretical Computer Science*, 2016(5), 2016. arXiv:1506.07429.

16 Laura Mančinska, David E. Roberson, and Antonios Varvitsiotis. Semidefinite relaxations of (quantum) graph isomorphism. Available online, 2017.

17 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014. arXiv:1301.1493.

18 Benjamin Musto and Jamie Vicary. Quantum Latin squares and unitary error bases, 2015. arXiv:1504.02715.

19 Ryan O'Donnell, John Wright, Chenggang Wu, and Yuan Zhou. *Hardness of Robust Graph Isomorphism,Lasserre Gaps, and Asymmetry of Random Graphs*, pages 1659–1677. ACM/SIAM, 2014. arXiv:1401.2436, doi:10.1137/1.9781611973402.120.

20 Carlos M. Ortiz and Vern I. Paulsen. Quantum graph homomorphisms via operator systems. *Linear Algebra and its Applications*, 497:23–43, 2016. arXiv:1505.00483.

21 Vern I. Paulsen, Simone Severini, Daniel Stahlke, Ivan G. Todorov, and Andreas Winter. Estimating quantum chromatic numbers. *Journal of Functional Analysis*, 270(6):2188–2222, 2016. arXiv:1407.6918.

22 Motakuri V. Ramana, Edward R. Scheinerman, and Daniel Ullman. Fractional isomorphism of graphs. *Discrete Mathematics*, 132(1):247–265, 1994. doi:10.1016/0012-365X(94)90241-0.

23 David E. Roberson. *Variations on a Theme: Graph Homomorphisms*. PhD thesis, University of Waterloo, 2013.

24 David E. Roberson. Conic formulations of graph homomorphisms. *Journal of Algebraic Combinatorics*, pages 1–37, 2016. arXiv:1411.6723, doi:10.1007/s10801-016-0665-y.

25 Jamie Sikora and Antonios Varvitsiotis. Linear conic formulations for two-party correlations and values of nonlocal games. *Mathematical Programming*, pages 1–33, 2015. arXiv:1506.07297.

26 Edwin R. van Dam and Willem H. Haemers. Which graphs are determined by their spectrum? *Linear Algebra and its Applications*, 373:241–272, 2003. doi:10.1016/S0024-3795(03)00483-X.

Honest Signaling in Zero-Sum Games Is Hard, and Lying Is Even Harder*†

Aviad Rubinstein

UC Berkeley, Berkeley, CA, USA
aviad@eecs.berkeley.edu

Abstract

We prove that, assuming the exponential time hypothesis, finding an ϵ-approximately optimal signaling scheme in a two-player zero-sum game requires quasi-polynomial time $(n^{\tilde{\Omega}(\lg n)})$. This is tight by [8] and resolves an open question of Dughmi [12]. We also prove that finding a multiplicative approximation is NP-hard.

We also introduce a new model where a dishonest signaler may publicly commit to use one scheme, but post signals according to a different scheme. For this model, we prove that even finding a $(1 - 2^{-n})$-approximately optimal scheme is NP-hard.

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

Keywords and phrases Signaling, Zero-sum Games, Algorithmic Game Theory, birthday repetition

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.77

1 Introduction

Many classical questions in economics involve extracting information from strategic agents. Lately, there has been growing interest within algorithmic game theory in *signaling*: the study of how to *reveal information* to strategic agents (see e.g. [16, 13, 14, 12, 8] and references therein). Signaling has been studied in many interesting economic and game theoretic settings. Among them, Zero-Sum Signaling proposed by Dughmi [12] stands out as a canonical problem that cleanly captures the computational nature of signaling. In particular, focusing on zero-sum games clears away issues of equilibrium selection and computational tractability of finding an equilibrium.

▶ **Definition 1** (Zero-Sum Signaling [12])**.** Alice and Bob play a Bayesian zero-sum game where the payoff matrix M is drawn from a publicly known prior. The signaler Sam privately observes the state of nature (i.e. the payoff matrix), and then publicly broadcasts a signal $\varphi(M)$ to both Alice and Bob. Alice and Bob Bayesian-update their priors according to $\varphi(M)$'s and play the Nash equilibrium of the expected game; but they receive payoffs according to the true M. Sam's goal is to design an efficient signaling scheme φ (a function from payoff matrices to strings) that maximizes Alice's expected payoff.

Dughmi's [12] main result proves that assuming the hardness of the Planted Clique problem, there is no additive FPTAS for Zero-Sum Signaling. The main open question

* A full version of the paper is available at http://arxiv.org/abs/1510.04991.

† This research was supported by Microsoft Research PhD Fellowship, as well as NSF grant CCF1408635 and by Templeton Foundation grant 3966. This work was done in part at the Simons Institute for the Theory of Computing.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 77; pp. 77:1–77:13

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

left by [12] is whether there exists an additive PTAS. Here we answer this question in the negative: we prove that assuming the Exponential Time Hypothesis (ETH) [15], obtaining an additive-ϵ-approximation (for some constant $\epsilon > 0$) requires quasi-polynomial time $(n^{\tilde{\Omega}(\lg n)})$. This result is tight thanks to a recent quasi-polynomial $(n^{\frac{\lg n}{\text{poly}(\epsilon)}})$ time algorithm by Cheng et al. [8]. Another important advantage of our result is that it replaces the hardness of Planted Clique with a more believable worst-case hardness assumption (see e.g. the discussion in [7]).

▶ **Theorem 2** (Main Result). *There exists a constant $\epsilon > 0$, such that assuming ETH, approximating* Zero-Sum Signaling *with payoffs in $[-1, 1]$ to within an additive ϵ requires time $n^{\tilde{\Omega}(\lg n)}$.*

Using a similar construction, we also obtain NP-hardness for computing a multiplicative-$(1-\epsilon)$-approximation. Unfortunately, in our example Alice can receives both negative and positive payoffs, which is somewhat non-standard (but not unprecedented [9]) in multiplicative approximation. One main reason that multiplicative approximation with negative payoffs is problematic is that this is often trivially intractable for any finite factor: Start with a tiny additive gap, where Alice's expected payoff is c in the "yes" case, and $s = c - \epsilon$ in the "no" case; subtract $(c + s)/2$ from all of Alice's payoffs to obtain an infinite multiplicative hardness. We note, however, that the combination of negative and positive payoffs in our construction serves only to obtain structural constraints on the resulting equilibria; the hardness of approximation is not a result of cancellation of negative with positive payoffs: Alice's payoff can be decomposed as a difference of non-negative payoffs $U = U^+ - U^-$, such that it is hard to approximate Alice's optimal payoff to within $\epsilon \cdot \mathsf{E}[U^+ + U^-]$. Nevertheless, we believe that extending this result to non-negative payoffs could be very interesting.

▶ **Theorem 3.** *There exists a constant $\epsilon > 0$, such that it is* NP*-hard to approximate* Zero-Sum Signaling *to within a multiplicative $(1-\epsilon)$ factor.*

Finally, we note that since all our games are zero-sum, the hardness results for Zero-Sum Signaling also apply to the respective notions of additive- and multiplicative-ϵ-Nash equilibrium.

1.1 The computational complexity of lying

As a motivating example, consider the purchase of a used car (not a zero-sum game, but a favorite setting in the study of signaling since Akerlof's seminal "Market for Lemons" [2]), and let us focus on the information supplied by a third party such as a mechanic inspection. The mechanic (Sam) publishes a signaling scheme: report any problem found in a one-hour inspection. Unbeknownst to the buyer (Bob), the mechanic favors the seller (Alice), and chooses to use a different signaling scheme: always report that the car is in excellent condition. Notice that it is crucial that the buyer does not know that the mechanic is lying (and more generally, we assume that neither party knows that the signaler is lying).

Much of the work in economics is motivated by selfish agents manipulating their private information. Here we introduce a natural extension of Dughmi's signaling model, where the signaler manipulates his private information. We formalize this extension in the Zero-Sum Lying problem, where the signaling scheme consists of two functions φ_{ALLEGED} ("report any problem found") and φ_{REAL} ("car is in excellent condition") from payoff matrices to signals. Sam promises Alice and Bob to use φ_{ALLEGED}, which is what Alice and Bob use to compute the posterior distribution given the signal (i.e. the seller and buyer look at the mechanic's

report and negotiate a price as if the state of the car is correctly reflected). But instead Sam signals according to φ_{REAL}.

We formally define the Zero-Sum Lying problem below; notice that the original Zero-Sum Signaling (Definition 1) corresponds to the special case where we restrict $\varphi_{\text{REAL}} = \varphi_{\text{ALLEGED}}$.

▶ **Definition 4** (Zero-Sum Lying)**.** Alice and Bob play a Bayesian, one-shot, zero-sum game where the payoff matrix is drawn from a publicly known prior. A *dishonest signaling scheme* consists of two (possibly randomized) functions $\varphi_{\text{ALLEGED}}, \varphi_{\text{REAL}}$ from payoff matrices to signals, that induce the following protocol:

- Nature draws a private payoff matrix $M \sim \mathcal{D}_{\text{NATURE}}$.
- Alice and Bob observe the scheme φ_{ALLEGED} and the signal $\sigma \triangleq \varphi_{\text{REAL}}(M)$. (But they don't know the scheme φ_{REAL}!)
- Alice and Bob choose a Nash equilibrium $(\mathbf{x}; \mathbf{y})$ for the zero-sum game with payoff matrix $\mathsf{E}[M' \mid \varphi_{\text{ALLEGED}}(M') = \sigma]$[1].
  - (We assume that the support of φ_{REAL} is contained in the support of φ_{ALLEGED}.)
- Alice and Bob receive payoffs $\mathbf{x}^\top M \mathbf{y}$ and $-\mathbf{x}^\top M \mathbf{y}$, respectively.

Sam's goal is to compute a pair $(\varphi_{\text{ALLEGED}}, \varphi_{\text{REAL}})$ that maximizes Alice's expected payoff.

In the toy-setting of a biased car inspection, the Sam's optimal strategy was very simple. In contrast, we show that for a general distribution over zero-sum games, it is NP-hard to find a pair $(\varphi_{\text{ALLEGED}}, \varphi_{\text{REAL}})$ that is even remotely close to optimal. Notice that this is very different from the honest case where, as we mentioned earlier, NP-hardness of additive approximation is unlikely given the additive quasi-PTAS of [8].

▶ **Theorem 5.** *Approximating* Zero-Sum Lying *with Alice's payoffs in* $[0, 1]$ *to within an additive* $(1 - 2^{-n})$ *is* NP*-hard.*

Further discussion of dishonest signaling

It is important to note that the dishonest signaling model has a few weaknesses:

- Alice and Bob must believe the dishonest signaler. (See also further discussion below.)
- In particular, Sam cheats in favor of Alice, but Alice doesn't know about it – so what's in it for Sam? Indeed, we assume that Sam has some intrinsic interest in Alice winning, e.g. because Sam loves Alice or owns some of her stocks.
- The game for which players' strategies are at equilibrium may be very different from the actual game. Note, however, that this is also the case for the honest signaling model (when the signaling scheme is not one-to-one).
- The players may receive different payoffs for different equilibria; this may raise issues of equilibrium selection.

Despite those disadvantages, we believe that our simple model is valuable because it already motivates surprising results such as our Theorem 5. On a higher level, we hope that it will inspire research on many other interesting aspects on dishonest signaling. For example, notice that in our model Sam lies without any reservation; if, per contra, the game was repeated infinitely many times, one would expect that Alice and Bob will eventually stop

[1] When $\varphi_{\text{ALLEGED}}, \varphi_{\text{REAL}}$ are randomized, we have $\sigma \sim \varphi_{\text{REAL}}(M)$ and expectation conditioned on $\mathsf{E}\left[M' \mid \sigma \sim \varphi_{\text{ALLEGED}}\left(M'\right)\right]$.

believing the signals, hence only honest signaling is possible. There is also a spectrum of intermediate situations, where Alice and Bob observe some partial information about past games (e.g. marginal distribution of signals) and may encounter questions about distribution testing.

Another related direction of potential future research is to think about Sam's incentives. When is honest signaling optimal for Sam? When is it approximately optimal? How should one design an effective "punishing" mechanism?

1.2 Concurrent work of Bhaskar et al.

In independent concurrent work by Bhaskar et al. [5], quasi-polynomial time hardness for additive approximation of Zero-Sum Signaling was obtained assuming the hardness of the Planted Clique problem (among other interesting results[2] about network routing games and security games). Although we are not aware of a formal reduction, hardness of Planted Clique is a qualitatively stronger assumption than ETH in the sense that it requires average case instances to be hard. Hence in this respect, our result is stronger.

1.3 Techniques

Our main ingredient for the quasi-polynomial hardness is the technique of "birthday repetition" coined by [1] and recently applied in game theoretic settings in [7, 4]: We reduce from a 2-ary constraint satisfaction problem (2-CSP) over n variables to a distribution over N zero-sum $N \times N$ games, with $N = 2^{\Theta(\sqrt{n})}$. Alice and Bob's strategies correspond to assignments to tuples of $\sqrt{n}$ variables. By the birthday paradox, the two $\sqrt{n}$-tuples chosen by Alice and Bob share a constraint with constant probability. If a constant fraction of the constraints are unsatisfiable, Alice's payoff will suffer with constant probability. Assuming ETH, approximating the value of the CSP requires time $2^{\tilde{\Omega}(n)} = N^{\tilde{\Omega}(\lg N)}$.

1.3.0.1 The challenge

The main difficulty is that once the signal is public, the zero-sum game is tractable. Thus we would like to force the signaling scheme to output a satisfying assignment. Furthermore, if the scheme would output partial assignments on different states of nature (aka different zero-sum games in the support), it is not clear how to check consistency between different signals. Thus we would like each signal to contain an entire satisfying assignment. The optimal scheme may be very complicated and even require randomization, yet by an application of the Caratheodory Theorem the number of signals is, wlog, bounded by the number of states of nature [12]. If the state of nature can be described using only $\lg N = \tilde{\Theta}(\sqrt{n})$ bits[3], how can we force the scheme to output an entire assignment?

To overcome this obstacle, we let the state of nature contain a partial assignment to a random $\sqrt{n}$-tuple of variables. We then check the consistency of Alice's assignment with nature's assignment, Bob's assignment with nature's assignment, and Alice and Bob's assignments with each other; let $\tau^{A,Z}, \tau^{B,Z}, \tau^{A,B}$ denote the outcomes of those consistency checks, respectively. Alice's payoff is given by:

$$U = \delta\tau^{A,Z} - \delta^2\tau^{B,Z} + \delta^3\tau^{A,B}$$

[2] For zero-sum games, Bhaskar et al. also rule out an additive FPTAS assuming $\mathsf{P} \neq \mathsf{NP}$. This result follows immediately from our Theorem 14.

[3] In other words, N, the final size of the reduction, is an upper bound on the number of states of nature.

for some small constant $\delta \in (0,1)$. Now, both Alice and Bob want to maximize their chances of being consistent with nature's partial assignment, and the signaling scheme gains by maximizing $\tau^{A,B}$.

Of course, if nature outputs a random assignment, we have no reason to expect that it can be completed to a full satisfying assignment. Instead, the state of nature consists of N assignments, and the signaling scheme helps Alice and Bob play with the assignment that can be completed.

Several other obstacles arise; fortunately some can be handled using techniques from previous works on hardness of finding Nash equilibrium [3, 10, 4].

2 Preliminaries

Exponential Time Hypothesis

▶ **Hypothesis 6** (Exponential Time Hypothesis (ETH) [15]). *3SAT takes time* $2^{\Omega(n)}$.

PCP Theorem and CSP

▶ **Definition 7** (2CSP). 2-CSP (2-ary Constraint Satisfaction Problem) is a maximization problem. The input is a graph $G=(V,E)$, alphabet Σ, and a constraint $C_e \subseteq \Sigma \times \Sigma$ for every $e \in E$.

The output is a labeling $\varphi : V \to \Sigma$ of the vertices. Given a labeling, we say that a constraint (or edge) $(u,v) \in E$ is *satisfied* if $\varphi(u), \varphi(v) \in C_{(u,v)}$. The *value of a labeling* is the fraction of $e \in E$ that are satisfied by the labeling. The value of the instance is the maximum fraction of constraints satisfied by any assignment.

▶ **Theorem 8** (PCP Theorem [11]; see e.g. [6, Theorem 2.11] for this formulation). *Given a 3SAT instance ϕ of size n, there is a polynomial time reduction that produces a 2CSP instance ψ, with size $|\psi| = n \cdot \mathrm{polylog} n$ variables and constraints, and constant alphabet size, such that:*

Completeness. *If ϕ is satisfiable, then so is ψ.*

Soundness. *If ϕ is not satisfiable, then at most a $(1-\eta)$-fraction of the constraints in ψ can be satisfied, for some $\eta = \Omega(1)$.*

Balance. *Every variable in ψ participates in exactly $d = O(1)$ constraints.*

Finding a good partition

▶ **Lemma 9** (Essentially [4, Lemma 6]). *Let $G=(V,E)$ be a d-regular graph and $n \triangleq |V|$. We can partition V into n/k disjoint subsets $\{S_1, \ldots, S_{n/k}\}$ of size at most $2k$ such that:*

$$\forall i,j \quad |(S_i \times S_j) \cap E| \leq 8d^2k^2/n\,. \tag{1}$$

See full version for proof [17].

How to catch a far-from-uniform distribution

The following lemma due to [10] implies that:

▶ **Lemma 10** (Lemma 3 in the full version of [10]). *Let $\{a_i\}_{i=1}^n$ be real numbers satisfying the following properties for some $\theta > 0$: (1) $a_1 \geq a_2 \geq \cdots \geq a_n$; (2) $\sum a_i = 0$; (3) $\sum_{i=1}^{n/2} a_i \leq \theta$. Then $\sum_{i=1}^n |a_i| \leq 4\theta$.*

3 Additive hardness

▶ **Theorem 11.** *There exists a constant $\epsilon > 0$, such that assuming ETH, approximating* Zero-Sum Signaling *with payoffs in $[-1, 1]$ to within an additive ϵ requires time $n^{\tilde{\Omega}(\lg n)}$.*

Construction overview

Our reduction begins with a 2CSP ψ over n variables from alphabet Σ. We partition the variables into n/k disjoint subsets $\{S_1, \ldots, S_{n/k}\}$, each of size at most $2k$ for $k = \sqrt{n}$ such that every two subsets share at most a constant number of constraints.

Nature chooses a random subset S_i from the partition, a random assignment $\vec{u} \in \Sigma^{2k}$ to the variables in S_i, and an auxiliary vector $\hat{b} \in \{0,1\}^{\Sigma \times [2k]}$. As mentioned in Section 1.3, $\vec{u}$ may not correspond to any satisfying assignment. Alice and Bob participate in one of $|\Sigma|^{2k}$ subgames; for each $\vec{v} \in \Sigma^{2k}$, there is a corresponding subgame where all the assignments are XOR-ed with $\vec{v}$. The goal of the auxiliary vector $\hat{b}$ is to force Alice and Bob to participate in the right subgame, i.e. the one where the XOR of $\vec{v}$ and $\vec{u}$ can be completed to a full satisfying assignment. In particular, the optimum signaling scheme reveals partial information about $\hat{b}$ in a way that guides Alice and Bob to participate in the right subgame. The scheme also outputs the full satisfying assignment, but reveals no information about the subset S_i chosen by nature.

Each player has $\left(|\Sigma|^{2k} \times 2\right) \times \left(n/k \times \binom{n/k}{n/2k} \times |\Sigma|^{2k}\right) = 2^{\Theta\left(\sqrt{n}\right)}$ strategies. The first $|\Sigma|^{2k}$ strategies correspond to a Σ-ary vector $\vec{v}$ that the scheme will choose after observing the random input. The signaling scheme forces both players to play (w.h.p.) the strategy corresponding to $\vec{v}$ by controlling the information that corresponds to the next 2 strategies. Namely, for each $\vec{v}' \in \Sigma^{2k}$, there is a random bit $b(\vec{v}')$ such that each player receives a payoff of 1 if they play $(\vec{v}', b(\vec{v}'))$ and 0 for $(\vec{v}', 1 - b(\vec{v}'))$. The b's are part of the state of nature, and the optimal signaling scheme will reveal only the bit corresponding to the special $\vec{v}$. Since there are $|\Sigma|^{2k}$ bits, nature cannot choose them independently, as that would require $2^{|\Sigma|^{2k}}$ states of nature. Instead we construct a pairwise independent distribution.

The next n/k strategies correspond to the choice of a subset S_i from the specified partition of variables. The $\binom{n/k}{n/2k}$ strategies that follow correspond to a gadget due to Althofer [3] whereby each player forces the other player to randomize (approximately) uniformly over the choice of subset.

The last $|\Sigma|^{2k}$ strategies correspond to an assignment to S_i. The assignment to each S_i is XOR-ed entry-wise with $\vec{v}$. Then, the players are paid according to checks of consistency between their assignments, and a random assignment to a random S_i picked by nature. (The scheme chooses $\vec{v}$ so that nature's random assignment is part of a globally satisfying assignment.) Each player wants to pick an assignment that passes the consistency check with nature's assignment. Alice also receives a small bonus if her assignment agrees with Bob's; thus her payoff is maximized when there exists a globally satisfying assignment.

See formal construction below, or refer to summary table in full version [17].

Formal construction

Let ψ be a 2CSP-d over n variables from alphabet Σ, as guaranteed by Theorem 8. In particular, ETH implies that distinguishing between a completely satisfiable instance and $(1-\eta)$-satisfiable requires time $2^{\tilde{\Omega}(n)}$. By Lemma 9, we can (deterministically and efficiently) partition the variables into n/k subsets $\{S_1, \ldots, S_{n/k}\}$ of size at most $2k = 2\sqrt{n}$, such that every two subsets share at most $8d^2k^2/n = O(1)$ constraints.

States of nature. Nature chooses a state $\left(\hat{b}, i, \vec{u}\right) \in \{0,1\}^{\Sigma\times[2k]} \times [n/k] \times \Sigma^{2k}$ uniformly at random. For each $\vec{v}$, $b(\vec{v})$ is the XOR of bits from $\hat{b}$ that correspond to entries of $\vec{v}$:

$$\forall \vec{v} \in \Sigma^{2k} \;\; b(\vec{v}) \triangleq \left(\bigoplus_{(\sigma,\ell):\ [\vec{v}]_\ell = \sigma} \left[\hat{b}\right]_{(\sigma,\ell)} \right).$$

Notice that the $b(\vec{v})$'s are pairwise independent and each marginal distribution is uniform over $\{0,1\}$.

Strategies. Alice and Bob each choose a strategy $(\vec{v}, c, j, T, \vec{w}) \in \Sigma^{2k} \times \{0,1\} \times [n/k] \times \binom{[n/k]}{n/2k} \times \Sigma^{2k}$. We use $\vec{v}^A, c^A$, etc. to denote the strategy Alice plays, and similarly $\vec{v}^B, c^B$, etc. for Bob. For $\sigma, \sigma' \in \Sigma$, we denote $\sigma \oplus_\Sigma \sigma' \triangleq \sigma + \sigma' \pmod{|\Sigma|}$, and for vectors $\vec{v}, \vec{v}' \in \Sigma^{2k}$, we let $\vec{v} \oplus_\Sigma \vec{v}' \in \Sigma^{2k}$ denote the entry-wise $\oplus_\Sigma$.

Payoffs. Consider state of nature $\left(\hat{b}, i, \vec{u}\right)$ and players' strategies $\left(\vec{v}^A, c^A, j^A, T^A, \vec{w}^A\right)$ and $\left(\vec{v}^B, c^B, j^B, T^B, \vec{w}^B\right)$.

When $\vec{v}^A = \vec{v}^B = \vec{v}$, we set $\tau^{A,Z} = 1$ if assignments $\vec{w}^A$ and $(\vec{v} \oplus_\Sigma \vec{u})$ to subsets S_{j^A} and S_i, respectively, satisfy all the constraints in ψ that are determined by $\left(S_i \cup S_{j^A}\right)$, and $\tau^{A,Z} = 0$ otherwise. Similarly, $\tau^{B,Z} = 1$ iff $\vec{w}^B$ and $(\vec{v} \oplus_\Sigma \vec{u})$ satisfy the corresponding constraints in ψ; and $\tau^{A,B}$ checks $\vec{w}^A$ and $\vec{w}^B$. When $\vec{v}^A \neq \vec{v}^B$, we set $\tau^{A,Z} = \tau^{B,Z} = \tau^{A,B} = 0$.

We decompose Alice's payoff as:

$$U^A \triangleq U^A_b + U^A_{\text{Althofer}} + U^A_\psi,$$

where

$$U^A_b \triangleq \mathbf{1}\left\{c^A = b\left(\vec{v}^A\right)\right\} - \mathbf{1}\left\{c^B = b\left(\vec{v}^B\right)\right\},$$

$$U^A_{\text{Althofer}} \triangleq \mathbf{1}\left\{j^B \in T^A\right\} - \mathbf{1}\left\{j^A \in T^B\right\},$$

and

$$U^A_\psi \triangleq \delta\tau^{A,Z} - \delta^2\tau^{B,Z} + \delta^3\tau^{A,B}, \tag{2}$$

for a sufficiently small constant $0 < \delta \ll \sqrt{\eta}$.

Completeness

▶ **Lemma 12.** *If ψ is satisfiable, there exists a signaling scheme and a mixed strategy for Alice that guarantees expected payoff $\delta - \delta^2 + \delta^3$.*

Proof. Fix a satisfying assignment $\vec{\alpha} \in \Sigma^n$. Given state of nature $\left(\hat{b}, i, \vec{u}\right)$, let $\vec{v}$ be such that $(\vec{v} \oplus_\Sigma \vec{u}) = [\vec{\alpha}]_{S_i}$. The scheme outputs the signal $(\vec{v}, b(\vec{v}), \vec{\alpha})$. Alice's mixed strategy sets $\left(\vec{v}^A, c^A\right) = (\vec{v}, b(\vec{v}))$, picks j^A and T^A uniformly at random, and sets $\vec{w}^A = [\vec{\alpha}]_{S_{j^A}}$.

Because Bob has no information about $b(\vec{v}')$ for any $\vec{v}' \neq \vec{v}$, he has probability $1/2$ of losing whenever he picks $\vec{v}^B \neq \vec{v}$, i.e. $\mathsf{E}\left[U^A_b\right] \geq \frac{1}{2}\Pr\left[\vec{v}^B \neq \vec{v}\right]$. Furthermore, because Alice chooses T^A and j^A uniformly, $\mathsf{E}\left[U^A_{\text{Althofer}}\right] = 0$.

Since $\vec{\alpha}$ completely satisfies ψ, we have that $\tau^{A,Z} = 1$ as long as $\vec{v}^B = \vec{v}$ (regardless of the rest of Bob's strategy). Bob's goal is thus to maximize $\mathsf{E}\left[\delta^2\tau^{B,Z} - \delta^3\tau^{A,B}\right]$. Notice that $\vec{w}^A$ and $(\vec{v} \oplus_\Sigma \vec{u})$ are two satisfying partial assignments to uniformly random subsets from the

partition. In particular, they are both drawn from the same distribution, so we have that for any mixed strategy that Bob plays, $\mathsf{E}\left[\tau^{B,Z}\right] = \mathsf{E}\left[\tau^{A,B}\right]$. Therefore Alice's payoff is at least

$$\left(\delta - \delta^2 + \delta^3\right) \Pr\left[\vec{v}^B = \vec{v}\right] + \frac{1}{2} \Pr\left[\vec{v}^B \neq \vec{v}\right] \geq \delta - \delta^2 + \delta^3. \qquad \blacktriangleleft$$

Soundness

▶ **Lemma 13.** *If at most a $(1-\eta)$-fraction of the constraints are satisfiable, Alice's maxmin payoff is at most $\delta - \delta^2 + (1 - \Omega_\eta(1))\,\delta^3$, for any signaling scheme.*

Proof. Fix any mixed strategy by Alice; we show that Bob can guarantee a payoff of at least $-\left(\delta - \delta^2 + (1 - \Omega_\eta(1))\,\delta^3\right)$. On any signal, Bob chooses $\left(\vec{v}^B, c^B\right)$ from the same distribution that Alice uses for $\left(\vec{v}^A, c^A\right)$. He chooses j^B uniformly, and picks T^B so as to minimize $\mathsf{E}\left[U^A_{\text{Althofer}}\right]$. Finally, for each j^B, he draws $\vec{w}^B$ from the same marginal distribution that Alice uses for $\vec{w}^A$ conditioning on $j^A = j^B$ (and uniformly at random if Alice never plays $j^A = j^B$). By symmetry, $\mathsf{E}\left[U^A_b\right] = 0$ and $\mathsf{E}\left[U^A_{\text{Althofer}}\right] \leq 0$.

In this paragraph, we use Althoefer's gadget to argue that, wlog, Alice's distribution over the choice of j^A is approximately uniform. In Althofer's gadget, Alice can guarantee an (optimal) expected payoff of 0 by randomizing uniformly over her choice of j^A and T^A. By Lemma 10, if Alice's marginal distribution over the choice of j^A is $8\delta^2$-far from uniform (in total variation distance), then Bob can guess that j^A is in some subset $T^B \in \binom{[n/k]}{n/2k}$ with advantage (over guessing at random) of at least $2\delta^2$. Therefore $\mathsf{E}\left[U^A_{\text{Althofer}}\right] \leq -2\delta^2$; but this would imply $\mathsf{E}\left[U^A\right] \leq -2\delta^2 + \mathsf{E}\left[U^A_\psi\right] \leq \delta - 2\delta^2 + \delta^3$. So henceforth we assume wlog that Alice's marginal distribution over the choice of j^A is $O\left(\delta^2\right)$-close to uniform (in total variation distance).

Since Alice's marginal distribution over j^A is $O\left(\delta^2\right)$-close to uniform, we have that Bob's distribution over $\left(j^B, \vec{w}^B\right)$ is $O\left(\delta^2\right)$-close to Alice's distribution over $\left(j^A, \vec{w}^A\right)$. Therefore $\mathsf{E}\left[\tau^{B,Z}\right] \geq \mathsf{E}\left[\tau^{A,Z}\right] - O\left(\delta^2\right)$, and so we also get:

$$\mathsf{E}\left[U^A\right] \leq \mathsf{E}\left[U^A_\psi\right] \leq \delta - \delta^2 + \delta^3 \mathsf{E}\left[\tau^{A,B}\right] + O\left(\delta^4\right). \tag{3}$$

Bounding $\mathsf{E}\left[\tau^{A,B}\right]$. To complete the proof, it remains to show an upper bound on $\mathsf{E}\left[\tau^{A,B}\right]$. In particular, notice that it suffices to bound the probability that Alice's and Bob's induced assignments agree. Intuitively, if they gave assignments to uniformly random (and independent) subsets of variables, the probability that their assignments agree cannot be much higher than the value of the 2CSP; below we formalize this intuition.

By the premise, any assignment to all variables violates at least an η-fraction of the constraints. In particular, this is true in expectation for assignments drawn according to Alice's and Bob's mixed strategy. This is a bit subtle: in general, it is possible that Alice's assignment alone doesn't satisfy many constraints and neither does Bob's, but when we check constraints between Alice's and Bob's assignments everything satisfied (for example, think of the 3-Coloring CSP, where Alice colors all her vertices blue, and Bob colors all his vertices red). Fortunately, this subtlety is irrelevant for our construction since we explicitly defined Bob's mixed strategy so that conditioned on each set S_j of variables, Alice and Bob have the same distribution over assignments.

The expected number of violations between pairs directly depends on the value of the 2CSP. To bound the *probability* of observing at least one violations, recall that every pair of subsets shares at most a constant number of constraints, so this probability is within a

constant factor of the expected number of violations. In particular, an $\Omega(\eta)$-fraction of the pairs of assignments chosen by Alice and Bob violate ψ.

Finally, Alice doesn't choose j^A uniformly at random; but her distribution is $O(\delta^2)$-close to uniform. Therefore, we have $\mathsf{E}\left[\tau^{A,B}\right] \leq 1 - \Omega(\eta) + O(\delta^2)$. Plugging into (3) completes the proof. ◀

4 Multiplicative hardness

▶ **Theorem 14.** *There exists a constant $\epsilon > 0$, such that it is* NP*-hard to approximate* Zero-Sum Signaling *to within a multiplicative $(1-\epsilon)$ factor.*

Construction overview

Our reduction begins with a 2CSP ψ over n variables from alphabet Σ.

Nature chooses a random index $i \in [n]$, a random assignment $u \in \Sigma$ for variable x_i, and an auxiliary vector $\vec{b} \in \{0,1\}^{\Sigma}$. Notice that u may not correspond to any satisfying assignment. Alice and Bob participate in one of $|\Sigma|$ subgames; for each $v \in \Sigma$, there is a corresponding subgame where all the assignments are XOR-ed with v. The optimum signaling scheme reveals partial information about $\vec{b}$ in a way that guides Alice and Bob to participate in the subgame where the XOR of v and u can be completed to a full satisfying assignment. The scheme also outputs the full satisfying assignment, but reveals no information about the index i chosen by nature.

Alice has $(|\Sigma| \times 2) \times (n \times n \times |\Sigma|) = \Theta(n^2)$ strategies, and Bob has an additional choice among n strategies (so $\Theta(n^3)$ in total). The first $|\Sigma|$ strategies correspond to a value $v \in \Sigma$ that the scheme will choose after observing the state of nature. The signaling scheme forces both players to play (w.h.p.) the strategy corresponding to v by controlling the information that corresponds to the next 2 strategies. Namely, for each $v' \in \Sigma$, there is a random bit $b(v')$ such that each player receives a small bonus if they play $(v', b(v'))$ and not $(v', 1 - b(v'))$. The b's are part of the state of nature, and the signaling scheme will reveal only the bit corresponding to the special v.

The next n strategies correspond to a choice of a variable $j \in [n]$. The n strategies that follow correspond to a hide-and-seek gadget whereby each player forces the other player to randomize (approximately) uniformly over the choice of j. For Bob, the additional n strategies induce a hide-and-seek game against nature, which serves to verify that the scheme does not reveal too much information about the state of nature (this extra verification was unnecessary in the reduction for additive inapproximability).

The last $|\Sigma|$ strategies induce an assignment for x_j. The assignment to each x_j is XOR-ed with v. Then, the players are paid according to checks of consistency between their assignments, and a random assignment to a random x_i picked by nature. (The scheme chooses v so that nature's random assignment is part of a globally satisfying assignment.) Each player wants to pick an assignment that passes the consistency check with nature's assignment. Alice also receives a small bonus if her assignment agrees with Bob's; thus her payoff is maximized when there exists a globally satisfying assignment.

Formal construction

Let ψ be a 2CSP-d over n variables from alphabet Σ, as guaranteed by Theorem 8. In particular, it is NP-hard to distinguish between ψ which is completely satisfiable, and one

where at most a $(1-\eta)$-fraction of the constraints can be satisfied. We denote $(i,j) \in \psi$ if there is a constraint over variables (x_i, x_j).

States of nature. Nature chooses a state $\left(\vec{b}, i, u\right) \in \{0,1\}^{\Sigma} \times [n] \times \Sigma$ uniformly at random.

Strategies. Alice chooses a strategy $\left(v^A, c^A, j^A, t^A, w^A\right) \in \Sigma \times \{0,1\} \times [n] \times [n] \times \Sigma$, and Bob chooses $\left(v^B, c^B, j^B, t^B, q^B, w^B\right) \in \Sigma \times \{0,1\} \times [n] \times [n] \times [n] \times \Sigma$. For $\sigma, \sigma' \in \Sigma$, we denote $\sigma \oplus_{\Sigma} \sigma' \triangleq \sigma + \sigma' \pmod{|\Sigma|}$, and for a vector $\vec{\alpha} \in \Sigma^n$ we let $(\sigma \oplus_{\Sigma} \vec{\alpha}) \in \Sigma^n$ denote the $\oplus_{\Sigma}$ of σ with each entry of $\vec{\alpha}$.

Payoffs. Consider players' strategies $\left(v^A, c^A, j^A, t^A, w^A\right)$ and $\left(v^B, c^B, j^B, t^B, q^B, w^B\right)$ and state of nature $\left(\vec{b}, i, u\right)$.

When $v^A = v^B = v$, we set $\tau^{A,Z} = 1$ if ψ contains a constraint for variables $\left(j^A, i\right)$, and the assignments w^A and $(v \oplus_{\Sigma} u)$ to those variables, respectively, satisfy this constraint, and $\tau^{A,Z} = 0$ otherwise. Similarly, $\tau^{B,Z} = 1$ iff w^B and $(v \oplus_{\Sigma} u)$ satisfy a corresponding constraint in ψ; and $\tau^{A,B}$ checks w^A with w^B. When $v^A \neq v^B$, we set $\tau^{A,Z} = \tau^{B,Z} = \tau^{A,B} = 0$.

We decompose Alice's payoff as:

$$U^A \triangleq U^A_b + U^A_{\text{seek}} + U^A_{\psi},$$

where

$$U^A_b \triangleq \mathbf{1}\left\{c^A = \left[\vec{b}\right]_{v^A}\right\}/n - \mathbf{1}\left\{c^B = \left[\vec{b}\right]_{v^B}\right\}/n,$$

$$U^A_{\text{seek}} \triangleq 2 \cdot \mathbf{1}\left\{j^B = t^A\right\} - \mathbf{1}\left\{j^A = t^B\right\} - \mathbf{1}\left\{i = q^B\right\},$$

and[4]

$$U^A_{\psi} \triangleq \delta^3 \tau^{A,Z} - \delta^4 \tau^{B,Z} + \delta^5 \tau^{A,B},$$

for a sufficiently small constant $0 < \delta \ll \sqrt{\eta}$.

Completeness

▶ **Lemma 15.** *If ψ is satisfiable, there exists a signaling scheme, such that for every signal $\mathbf{s}$ in the support, Alice can guarantee an expected payoff of $\frac{d}{n}\left(\delta^3 - \delta^4 + \delta^5\right)$.*

Notice that the *for every signal in the support* qualification is different than the corresponding Lemma 12 (and there is a similar difference between Lemma 16 and Lemma 13). Indeed, this is stronger than we need for proving Theorem 14, but will come handy in Section 5.

Proof. Fix a satisfying assignment $\vec{\alpha} \in \Sigma^n$. Given state of nature $\left(\hat{b}, i, u\right)$, let v be such that $(v \oplus_{\Sigma} u) = [\vec{\alpha}]_i$. The scheme outputs the signal $\mathbf{s} \triangleq \left(v, \vec{b}_v, \vec{\alpha}\right)$. Alice's mixed strategy sets $\left(v^A, c^A\right) = \left(v, \vec{b}_v\right)$; picks j^A and t^A uniformly at random; and sets $w^A = [\vec{\alpha}]_{j^A}$. See full version for details [17]. ◀

[4] We use $\delta^3\tau^{A,Z} - \delta^4\tau^{B,Z} + \delta^5\tau^{A,B}$ instead of $\delta^1\tau^{A,Z} - \delta^2\tau^{B,Z} + \delta^3\tau^{A,B}$ as in 2, because the square of the first coefficient appears in the proof. We have $\left(\delta^3\right)^2 \ll \delta^5$, but $\delta^2 \gg \delta^3$.

Soundness

▶ **Lemma 16.** *If at most a* $(1-\eta)$*-fraction of the constraints are satisfiable, then for any signaling scheme and every signal* $\mathbf{s}$ *in the support, Alice's maxmin payoff is at most* $\frac{d}{n}\left(\delta^3-\delta^4+\left(1-\Omega\left(1\right)\right)\delta^5\right)$.

Proof. On any signal, Bob chooses $\left(v^B,c^B\right)$ from the same distribution that Alice uses for $\left(v^A,c^A\right)$. He draws j^B uniformly at random, and picks t^B and q^B so as to minimize $\mathsf{E}\left[U^A_{\text{seek}}\mid\mathbf{s}\right]$. Finally, for each j^B, Bob draws w^B from the same distribution that Alice uses for w^A conditioning on $j^A=j^B$ (and uniformly at random if Alice never plays $j^A=j^B$). By symmetry, $\mathsf{E}\left[U^A_b\mid\mathbf{s}\right]=0$ and $\mathsf{E}\left[U^A_{\text{seek}}\mid\mathbf{s}\right]\leq 0$. See full version for details [17]. ◀

5 Lying is even harder

▶ **Theorem 17.** *Approximating* Zero-Sum Lying *with Alice's payoffs in* $[0,1]$ *to within an additive* $(1-2^{-n})$ *is* NP*-hard.*

Construction

Consider the construction from Section 4 for the honest signaling problem. Lemmata 15 and 16 guarantee that there exists a distribution $\mathcal{D}_{\text{HONEST}}$ of $n\times n$ zero-sum games and constants $c_1>c_2$ such that it is NP-hard to distinguish between the following:

Completeness. If ψ is satisfiable, there exists a signaling scheme φ_{HONEST}, such that for any signal in φ_{HONEST}'s support, Alice's maxmin payoff is at least c_1/n.

Soundness. If ψ is $(1-\eta)$-unsatisfiable, for every signaling scheme φ'_{HONEST} and every signal in the support, Alice's maxmin payoff is at most c_2/n.

For Zero-Sum Lying, we construct a hard distribution of $n\times(n+1)$ zero-sum games as follows. With probability 2^{-n} Alice's payoffs matrix is of the form:

$$\begin{pmatrix} & -\left(c_1+c_2\right)/2n \\ -A^{\top}_{\text{HONEST}} & \vdots \\ & -\left(c_1+c_2\right)/2n \end{pmatrix}, \tag{4}$$

where Alice chooses a row (Bob chooses a column), and A_{HONEST} is an $n\times n$ matrix drawn from $\mathcal{D}_{\text{HONEST}}$. In other words, Bob has to choose between receiving payoff $\left(c_1+c_2\right)/2n$, or playing a game drawn from $\mathcal{D}_{\text{HONEST}}$, but with the roles reversed.

Otherwise (with probability $1-2^{-n}$), Alice's payoff depends only on Bob: it is 1 if Bob chooses any of his first n actions, and 0 otherwise; we call this the *degenerate game*.

Notice that we promised payoffs in $[0,1]$, whereas (4) has payoffs in $[-1,0]$. $[0,1]$ payoffs can be obtained, without compromising the inapproximability guarantee, by scaling and shifting the entries in (4) in a straightforward manner.

Completeness

▶ **Lemma 18.** *If* ψ *is satisfiable, there exists a dishonest signaling scheme, such that Alice's expected payoff is at least* $1-2^{-n}$.

Proof. We first construct φ_{ALLEGED} as follows. Whenever nature samples a payoff matrix as in (4), φ_{ALLEGED} outputs the signal that φ_{HONEST} would output for A_{HONEST}. Whenever Alice and Bob play the degenerate game, φ_{ALLEGED} outputs a special symbol $\perp$.

When Bob observes any symbol from the support of φ_{HONEST}, he can guarantee a payoff of $c_1/n > (c_1 + c_2)/2n$ by playing a mix of his first n strategies. Therefore he only uses his last strategy when observing the special symbol $\perp$.

Our true signaling scheme φ_{REAL} always outputs an (arbitrary) signal from the support of φ_{HONEST}, regardless of the state of nature. With probability $1 - 2^{-n}$, Alice and Bob are actually playing the degenerate game, so Alice's payoff is 1. ◀

Soundness

▶ **Lemma 19.** *If ψ is $(1-\eta)$-unsatisfiable, then for any dishonest signaling scheme $\left(\varphi'_{\text{ALLEGED}}, \varphi'_{\text{REAL}}\right)$, Alice's expected payoff is negative.*

Proof. Any signal in the support of $\varphi'_{\text{ALLEGED}}$ corresponds to a mixture of the degenerate game, and the distribution induced by some signal $\mathbf{s}'$ in the support of some honest signaling scheme φ'_{HONEST} for $\mathcal{D}_{\text{HONEST}}$. In the degenerate game, Bob always prefers to play his last strategy. For any $\mathbf{s}'$, Bob again prefers a payoff of $(c_1 + c_2)/2n$ for playing his last strategy over a maxmin of at most c_2/n when playing any mixture of his first n strategies. Therefore, Bob always plays his last strategy, regardless of the signal he receives, which guarantees him a payoff of $(c_1 + c_2)/2^{n+1}n > 0$. ◀

Acknowledgements. I thank Shaddin Dughmi for explaining [8]. I thank Jonah Brown-Cohen, Rishi Gupta, Christos Papadimitriou, Tselil Schramm, and anonymous reviewers for helpful comments on earlier drafts.

References

1 Scott Aaronson, Russell Impagliazzo, and Dana Moshkovitz. AM with multiple merlins. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 44–55. IEEE, 2014.

2 George Akerlof. The market for lemons: Qualitative uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, 1970.

3 Ingo Althofer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199:339–355, 1994.

4 Yakov Babichenko, Christos H. Papadimitriou, and Aviad Rubinstein. Can almost everybody be almost happy? In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 1–9, 2016. doi:10.1145/2840728.2840731.

5 Umang Bhaskar, Yu Cheng, Young Kun Ko, and Chaitanya Swamy. Hardness results for signaling in bayesian zero-sum and network routing games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC'16, Maastricht, The Netherlands, July 24-28, 2016*, pages 479–496, 2016. doi:10.1145/2940716.2940753.

6 Mark Braverman, Young Kun-Ko, Aviad Rubinstein, and Omri Weinstein. ETH hardness for densest-k-subgraph with perfect completeness. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1326–1341, 2017. doi:10.1137/1.9781611974782.86.

7 Mark Braverman, Young Kun-Ko, and Omri Weinstein. Approximating the best Nash Equilibrium in $n^{o(\log n)}$-time breaks the Exponential Time Hypothesis. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*

2015, San Diego, CA, USA, January 4-6, 2015, pages 970–982, 2015. doi:10.1137/1.9781611973730.66.

8 Yu Cheng, Ho Yee Cheung, Shaddin Dughmi, Ehsan Emamjomeh-Zadeh, Li Han, and Shang-Hua Teng. Mixture selection, mechanism design, and signaling. In *FOCS*, 2015. To appear. URL: http://arxiv.org/pdf/1508.03679v1.pdf.

9 Constantinos Daskalakis. On the Complexity of Approximating a Nash Equilibrium. *ACM Transactions on Algorithms*, 9(3):23, 2013. doi:10.1145/2483699.2483703.

10 Constantinos Daskalakis and Christos H. Papadimitriou. On oblivious PTAS's for Nash equilibrium. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 75–84, 2009. Full version available at http://arxiv.org/abs/1102.2280. doi:10.1145/1536414.1536427.

11 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.

12 Shaddin Dughmi. On the hardness of signaling. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 354–363, 2014. doi:10.1109/FOCS.2014.45.

13 Shaddin Dughmi, Nicole Immorlica, and Aaron Roth. Constrained signaling for welfare and revenue maximization. *SIGecom Exchanges*, 12(1):53–56, 2013. doi:10.1145/2509013.2509022.

14 Yuval Emek, Michal Feldman, Iftah Gamzu, Renato Paes Leme, and Moshe Tennenholtz. Signaling schemes for revenue maximization. *ACM Trans. Economics and Comput.*, 2(2):5, 2014. doi:10.1145/2594564.

15 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

16 Peter Bro Miltersen and Or Sheffet. Send mixed signals: earn more, work less. In *ACM Conference on Electronic Commerce, EC'12, Valencia, Spain, June 4-8, 2012*, pages 234–247, 2012. doi:10.1145/2229012.2229033.

17 Aviad Rubinstein. Honest signaling in zero-sum games is hard, and lying is even harder. *CoRR*, abs/1510.04991, 2015. URL: http://arxiv.org/abs/1510.04991.

A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs*†

Pasin Manurangsi[1] and Prasad Raghavendra[2]

1 **University of California, Berkeley, CA, USA**
pasin@berkeley.edu
2 **University of California, Berkeley, CA, USA**
prasad@berkeley.edu

Abstract

A $(k \times l)$-*birthday repetition* $\mathcal{G}^{k\times l}$ of a two-prover game $\mathcal{G}$ is a game in which the two provers are sent random sets of questions from $\mathcal{G}$ of sizes k and l respectively. These two sets are sampled independently uniformly among all sets of questions of those particular sizes. We prove the following *birthday repetition theorem*: when $\mathcal{G}$ satisfies some mild conditions, $val(\mathcal{G}^{k\times l})$ decreases exponentially in $\Omega(kl/n)$ where n is the total number of questions. Our result positively resolves an open question posed by Aaronson, Impagliazzo and Moshkovitz [Aaronson et al., CCC, 2014].

As an application of our birthday repetition theorem, we obtain new fine-grained inapproximability results for dense CSPs. Specifically, we establish a tight trade-off between running time and approximation ratio by showing conditional lower bounds, integrality gaps and approximation algorithms; in particular, for any sufficiently large i and for every $k \geq 2$, we show the following:

- We exhibit an $O(q^{1/i})$-approximation algorithm for dense Max k-CSPs with alphabet size q via $O_k(i)$-level of Sherali-Adams relaxation.
- Through our birthday repetition theorem, we obtain an integrality gap of $q^{1/i}$ for $\tilde{\Omega}_k(i)$-level Lasserre relaxation for fully-dense Max k-CSP.
- Assuming that there is a constant $\varepsilon > 0$ such that Max 3SAT cannot be approximated to within $(1-\varepsilon)$ of the optimal in sub-exponential time, our birthday repetition theorem implies that any algorithm that approximates fully-dense Max k-CSP to within a $q^{1/i}$ factor takes $(nq)^{\tilde{\Omega}_k(i)}$ time, almost tightly matching our algorithmic result.

As a corollary of our algorithm for dense Max k-CSP, we give a new approximation algorithm for Densest k-Subhypergraph, a generalization of Densest k-Subgraph to hypergraphs. When the input hypergraph is $O(1)$-uniform and the optimal k-subhypergraph has constant density, our algorithm finds a k-subhypergraph of density $\Omega(n^{-1/i})$ in time $n^{O(i)}$ for any integer $i > 0$.

1998 ACM Subject Classification G.1.6 Linear Programming, G.2.2 Graph Algorithms

Keywords and phrases Birthday Repetition, Constraint Satisfaction Problems, Linear Program

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.78

1 Introduction

Polynomial-time reductions between computational problems are among the central tools in complexity theory. The rich and vast theory of hardness of approximation emerged out

* The full version of this extended abstract can be found at [42], https://arxiv.org/abs/1607.02986.
† This material is based upon work supported by NSF CCF-1343104 and the Okawa Research Grant.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 78; pp. 78:1–78:15


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

of the celebrated PCP Theorem [6] and the intricate web of polynomial-time reductions developed over the past two decades. During this period, an extensive set of reduction techniques such as parallel repetition and long-codes have been proposed and a variety of mathematical tools including discrete harmonic analysis, information theory and Gaussian isoperimetry have been applied towards analyzing these reductions. These developments have led to an almost complete understanding of the approximability of many fundamental combinatorial optimization problems like Set Cover and Max 3SAT. Yet, there are a few central problems such as computing approximate Nash equlibria, Densest k-Subgraph and Small Set Expansion, that remain out of reach of the web of polynomial-time reductions.

A promising new line of work proposes to understand the complexity of these problems through the lens of *sub-exponential time reductions*. Specifically, the idea is to construct a sub-exponential time reduction from 3SAT to the problem at hand, say, the Approximate Nash Equilibrium problem. Assuming that 3SAT does not admit sub-exponential time algorithms (also known as the Exponential Time Hypothesis (ETH) [35]), this would rule out polynomial time algorithms for the Approximate Nash Equilibrium problem.

At the heart of this line of works, lies the so-called *birthday repetition* of two-prover games. To elaborate on this, we begin by formally defining the notion of two-prover games.

▶ **Definition 1.** A *two-prover game* $\mathcal{G}$ consists of

- A finite set of questions X, Y and corresponding answer sets (aka alphabets) Σ_X, Σ_Y.
- A distribution $\mathcal{Q}$ over pairs of questions $X \times Y$.
- A verification function $P : X \times Y \times \Sigma_X \times \Sigma_Y \to \{0, 1\}$.

The value of $\mathcal{G}$ is the maximum over all strategies $\phi : X \cup Y \to \Sigma_X \cup \Sigma_Y$ of the output of P, i.e., $val(\mathcal{G}) = \max_{\phi: X\cup Y \to \Sigma_X \cup \Sigma_Y} \mathbb{E}_{(x,y)\sim\mathcal{Q}}[P(x, y, \phi(x), \phi(y))]$. We use n and q to denote the number of variables $|X| + |Y|$ and the alphabet size $|\Sigma_X| + |\Sigma_Y|$ respectively.

Two prover games earn their name from the following interpretation of the above definition: The game $\mathcal{G}$ is played between a verifier V and two cooperating provers $Merlin_1$ and $Merlin_2$ who have agreed upon a common strategy, but cannot communicate with each other during the game. The verifier samples two questions $(x, y) \sim \mathcal{Q}$ and sends x to $Merlin_1$ and y to $Merlin_2$. The provers respond with answers $\phi(x)$ and $\phi(y)$, which the verifier accepts or rejects based on the value of the verification function $P(x, y, \phi(x), \phi(y))$.

Two-prover games and, more specifically, a special class of two-prover games known as Label Cover are the starting points for reductions in a large body of hardness of approximation results. The PCP theorem implies that for some constant ε_0, approximating the value of a two prover game to within an additive ε_0 is **NP**-hard. However, this hardness result on its own is inadequate to construct reductions to other combinatorial optimization problems. To this end, this hardness result can be strengthened to imply that it is **NP**-hard to approximate the value of two-prover games to any constant factor, using the *parallel repetition theorem*.

For an integer k, the k-parallel repetition $\mathcal{G}^{\otimes k}$ of $\mathcal{G}$ can be described as follows. The question and answer sets in $\mathcal{G}^{\otimes k}$ consist of k-tuples of questions and answers from $\mathcal{G}$. The distribution over questions in $\mathcal{G}^{\otimes k}$ is given by the product distribution $\mathcal{Q}^k$. The verifier for $\mathcal{G}^{\otimes k}$ accepts the answers if and only if the verifier for $\mathcal{G}$ accepts each of the k individual answers.

Roughly speaking, the parallel repetition theorem asserts that the value of $\mathcal{G}^k$ decays exponentially in k. The theorem forms a key ingredient in obtaining hardness of approximation results, and have aptly received considerable attention in literature [51, 34, 50, 25, 46, 14].

Birthday repetition, introduced by Aaronson et al. [2], is an alternate transformation on two-prover games defined as follows.

▶ **Definition 2.** The $(k \times l)$-*birthday repetition* of a two-prover game $\mathcal{G}$ consists of

- The set of questions in $\mathcal{G}^{k\times l}$ are $\binom{X}{k}$ and $\binom{Y}{l}$ respectively, i.e., each question is a subset $S \subseteq X$ of size k and subset $T \subseteq Y$ of size l.
- The distribution over questions is the uniform product distribution over $\binom{X}{k} \times \binom{Y}{l}$.
- The verifier accepts only if, for every $(x, y) \in S \times T$ such that (x, y) form a valid pair of questions in $\mathcal{G}$, i.e., $(x, y) \in \text{supp}(\mathcal{Q})$, the answers to x and y are accepted $\mathcal{G}$.

The basic idea of birthday repetition can be traced back to the work of Aaronson et al. [1] on quantum multiprover proof systems $\mathbf{QMA}(k)$ for 3SAT. Subsequent work by Aaronson et al. [2] on the classical analogue of $\mathbf{QMA}(k)$ formally defined birthday repetition for two-prover games, and set the stage for applications in hardness of approximation.

Unlike parallel repetition, birthday repetition is only effective for large values of k and l. In particular, if $k, l < o(\sqrt{n})$, then, for most pairs of S and T, there is no $(x, y) \in S \times T$ such that (x, y) belongs to the support of the questions in the original game. However, if we pick $k = l = \omega(\sqrt{n})$, then by the birthday paradox, with high probability the sets S, T contain an edge (x, y) from the original game $\mathcal{G}$. Hence, for this choice of k and l, the game played by the provers is seemingly at least as difficult to succeed, as the original game $\mathcal{G}$. Aaronson et al. [2] confirmed this intuition by proving the following theorem.

▶ **Theorem 3** ([2]). *For any two-prover game $\mathcal{G}$ such that $\mathcal{Q}$ is uniform over its support, if the bipartite graph induced by $(X, Y, \text{supp}(\mathcal{Q}))$ is biregular, then $val(\mathcal{G}^{k\times l}) \leq val(\mathcal{G}) + O(\sqrt{\frac{n}{kl}})$.*

On the one hand, birthday repetition is ineffective in that it has to incur a blowup of $2^{\sqrt{n}}$ in the size, to even simulate the original game $\mathcal{G}$. The distinct advantage of birthday repetition is that the resulting game $\mathcal{G}^{k,l}$ has a distinct structure – in that it is a *free game*.

▶ **Definition 4.** A *free game* is a two-player game such that $\mathcal{Q}$ is uniform over $X \times Y$.

The birthday repetition theorem of [2] immediately implies a hardness of approximation for the value of free games. Specifically, they show that it is ETH-hard to approximate free games to some constant ratio in almost quasi-polynomial time. Interestingly, this lower bound is nearly tight in that free games admit a quasipolynomial time approximation scheme [10, 2].

Following Aaronson et al.'s work, birthday repetition has received numerous applications, which can be broadly classified in to two main themes. On the one hand, there are problems such as computing approximate Nash equilibria [16, 8, 54], approximating free games [2], approximating learning dimensions [43], and approximate symmetric signaling in zero sum games [53], where the underlying problems admit quasipolynomial-time algorithms [26, 38, 28] and birthday repetition can be used to show that such a running time is necessary, assuming ETH. On the other hand, there are computational problems like Densest k-Subgraph [15, 39], injective tensor norms [1, 33, 9], 2-to-4-norms [1, 33, 9] wherein an **NP**-hardness of approximation result seems out of reach of current techniques. But the framework of birthday repetition can be employed to show a quasi-polynomial hardness assuming ETH[1].

Unlike the parallel repetition theorem, the birthday repetition theorem of [2] does not achieve any reduction in the value of the game. It is thus natural to ask whether birthday repetition can also be used to decrease the value of a game. Aaronson et al. conjectured that the value of the birthday repetition game indeed deteriorates exponentially in $\Omega(kl/n)$, which is the expected number of edges between S and T in birthday repetition. Our main contribution is that we resolve the conjecture positively by showing the following theorem.

[1] Although the hardness results for injective tensor norms and 2-to-4-norms build over quantum multiprover proof systems, the basic idea of birthday repetition [1] lies at the heart of these reductions.

▶ **Theorem 5** (Birthday Repetition Theorem (informal)). *Let $\mathcal{G} = (X, Y, \mathcal{Q}, \Sigma_X, \Sigma_Y, P)$ be a two-prover game such that $\mathcal{Q}$ is uniform over its support, $(X, Y, \mathrm{supp}(\mathcal{Q}))$ is biregular and $|\Sigma_X|, |\Sigma_Y|$ are constant. If $val(\mathcal{G}) = 1 - \varepsilon$, then $val(\mathcal{G}^{k\times l}) \leq 2(1 - \varepsilon/2)^{\Omega(\varepsilon^5 kl/n)}$.*

Note that our result is more general than stated above and can handle irregular graphs and non-constant alphabet sizes as well (see Theorem 12 and Theorem 13).

By definition, our theorem immediately implies the following inapproximability of free games.

▶ **Corollary 6.** *Unless ETH is false, no polynomial time algorithm can approximate the value of a free game to within a factor of $2^{\tilde{\Omega}(\log(nq))}$.*

1.1 Dense CSPs

Free games can be viewed as 2-ary constraint satisfaction problems (CSP). From this perspective, free games are *dense*, in that there are constraints on a constant fraction of all pairs of variables. As an application of our birthday repetition theorem, we show almost-tight lower bounds for dense CSPs. To this end, we begin by defining CSPs and its density.

▶ **Definition 7.** A Max k-CSP instance $\mathcal{G}$ consists of

- A finite set of variables V and a finite alphabet set Σ.
- A distribution $\mathcal{Q}$ over k-tuple of variables V^k.
- A predicate $P : V^k \times \Sigma^k \to [0, 1]$.

Similar to two-prover games, $val(\mathcal{G})$ is defined as $\max_{\phi:V\to\Sigma} \mathbb{E}_{S\sim\mathcal{Q}}[P(S, \phi|_S)]$ where $\phi|_S$ is the restriction of the assignment to S, and we use n to denote the number of variables $|V|$ and q to denote the alphabet size $|\Sigma|$ of $\mathcal{G}$. Finally, $\mathcal{G}$ is called Δ-*dense* if $\Delta \cdot \mathcal{Q}(S) \leq 1/|V|^k$ for every $S \in V^k$. The 1-dense instances are also said to be fully-dense.

There has been a long line of works on approximating dense CSPs. Arora et al. were first to devise a polynomial-time approximation scheme for the problem when alphabet size is constant [5]. Since then, numerous algorithms have been invented for approximating dense CSPs using variety of techniques such as combinatorial algorithms with exhaustive sampling [5, 21, 44, 58, 40, 29], subsampling of instances [3, 10], regularity lemmas [30, 20] and LP/SDP hierarchies [22, 11, 31, 60]. Among the known algorithms, the fastest is Yaroslavtsev's [58] that achieves $(1 + \varepsilon)$-approximation in $q^{O_k(\log q)} + (nq)^{O(1)}$ time[2].

Unfortunately, when q is (almost-)polynomial in n, none of the above algorithms run in polynomial time. CSPs in such regime of parameters have long been studied in hardness of approximation (e.g. [12, 52, 7, 24, 47, 45]) and have recently received more attention from the approximation algorithm standpoint, both in the general case [48, 17, 41, 19] and the dense case [40]. The approximabilities of these two cases are vastly different. In the general case, approximating Max 2-CSP to within a factor of $2^{\log^{1-\varepsilon}(nq)}$ is **NP**-hard for any constant $\varepsilon > 0$ [24]. Moreover, the long-standing Sliding Scale Conjecture [12] states that this ratio can be improved to $(nq)^{\varepsilon}$ for some constant $\varepsilon > 0$. On the other hand, aforementioned algorithms for dense CSPs rule out such hardnesses for the dense case.

While the gap between known approximation algorithms and inapproximability results in the general case is tiny ($2^{\log^{\varepsilon}(nq)}$ for any constant $\varepsilon > 0$), the story is different for the dense case, especially when we restrict ourselves to polynomial-time algorithms. Aaronson et al. only ruled out, assuming ETH, polylog(nq)-approximation for such algorithms [2].

[2] [58] states that the algorithm takes $q^{O_k(1)} + (nq)^{O(1)}$ time, which is incorrect [59].

However, for $k > 2$, no non-trivial polynomial-time algorithm for dense Max k-CSP on large alphabet is even known. In this paper, we settle down the complexity of approximating dense Max k-CSP almost completely by answering the following fine-grained question: for each $i \in \mathbb{N}$, what is the best approximation for dense Max k-CSP, achievable in time $(nq)^i$?

Manurangsi and Moshkovitz developed an algorithm for dense Max 2-CSP that, when the instance has value $\Omega(1)$, obtains $O(q^{1/i})$-approximation in $(nq)^{O(i)}$ time [40]. Unfortunately, the algorithm does not work for dense Max k-CSP when $k > 2$. Using a conditioning-based rounding technique developed in [11, 49, 60], we show that the Sherali-Adams (SA) relaxation [56] exhibits a similar approximation even when $k > 2$, as stated below.

▶ **Theorem 8** (Informal). *For every $i > 0$ and any dense* Max k-CSP *instance of value $1-\varepsilon$, an $O_{k,\varepsilon}(i/\Delta)$-level of the SA relaxation yields an $O(q^{1/i})$-approximation for the instance.*

Using our birthday repetition theorem, we prove that the above tradeoff between run-time and approximation ratio cannot be improved even with the stronger Lasserre hierarchy [37]. Specifically, by applying the birthday repetition theorem with $k, l = \Omega(n \log i/i)$ on an $\Omega(n)$-level Lasserre integrality gap for Max 3XOR [55], we show the following.

▶ **Lemma 9** (Informal). *For every sufficiently large $i > 0$, there is a fully-dense* Max k-CSP *instance of value $1/(nq)^{1/i}$ such that the value of $\tilde{\Omega}_k(i)$-level Lasserre relaxation is one.*

Instead, if we assume that there exists a constant $\varepsilon > 0$ so that Max 3SAT cannot be approximated to $1 - \varepsilon$ in sub-exponential time (which we call the Exponential Time Hypothesis for Approximating 3SAT (ETHA)[3]), we can similarly arrive at the following hardness result.

▶ **Lemma 10** (Informal). *Unless ETHA is false, for every sufficiently large $i > 0$, no $(nq)^{\tilde{O}_k(i)}$-time algorithm approximates fully-dense* Max k-CSP *to within a factor of $(nq)^{1/i}$.*

Thus, assuming ETHA, our results resolve complexity of approximating dense CSPs up to a factor of polylog i and a dependency on k in the exponent of the running time.

1.2 Densest k-Subhypergraph

As a by-product of our algorithm for dense Max k-CSP, we give an approximation algorithm for the following Densest k-Subhypergraph problem: given a hypergraph (V, E), find $S \subseteq V$ of k vertices that maximizes the number of edges contained in S.

When the input hypergraph is simply a graph, the problem becomes Densest k-Subgraph, which has been extensively studied dating back to the early '90s [36, 27, 28, 57, 13]. On the other hand, Densest k-Subhypergraph was first studied in 2006, when Hajiaghayi et al. [32] proved that, if 3SAT $\notin$ **DTIME**$(2^{n^{3/4+\varepsilon}})$ for some $\varepsilon > 0$, then no polynomial-time algorithm approximates the problem to within a factor of $2^{\log^{\delta} n}$ for some $\delta > 0$. Later, Applebaum [4] showed, under a cryptographic assumption, that, for sufficiently large d, Densest k-Subhypergraph on d-uniform hypergraph is hard to approximate to a factor of n^{ε} for some $\varepsilon > 0$. More recently, Chlamtác et al. [18] provided the first non-trivial approximation algorithm for the problem; their algorithm works only on 3-uniform hypergraph and achieves $O(n^{4(4-\sqrt{3})/13+\varepsilon})$-approximation for any constant $\varepsilon > 0$ in polynomial time.

[3] ETHA is also introduced independently as gap-ETH by Dinur [23] who uses it to provide a supporting evidence to the Sliding Scale Conjecture.

Thanks to Charikar et al.'s [17] reduction from Densest k-Subgraph to Max 2-CSP, which can be adapted to reduce Densest k-Subhypergraph on d-uniform hypergraph to Max d-CSP, Theorem 8 implies the following algorithm for Densest k-Subhypergraph.

▶ **Corollary 11** (Informal). *There is a randomized algorithm that, given a d-uniform hypergraph on n vertices whose densest k-subhypergraph is Δ-dense and an integer $i > 0$, runs in $n^{O_d(i/\Delta)}$ time and outputs a k-subhypergraph of density $\Omega_k(\Delta/n^{1/i})$ with high probability.*

Here the density of a d-uniform hypergraph is defined as $d!|E|/|V|^d$. Note that the density condition required is on the optimum not the input. Moreover, when Δ and d are constant, the algorithm provides an $O(n^{1/i})$ approximation in $n^{O(i)}$ time for every $i > 0$. When $d = 2$, this matches the previously known algorithms for Densest k-Subgraph [28, 57, 40].

Organization of the Paper

In Section 2, we provide preliminaries and notations used in the paper. Then, in Section 3, we outline the proofs of our main theorems; the full proofs are deferred to the full version of this work [42]. Next, the algorithm for dense CSPs is described in Section 4. Finally, we conclude by proposing open questions in Section 5. Note that the lower bounds for dense CSPs and the algorithm for Densest k-Subhypergraph are also deferred to the full version.

2 Preliminaries and Notations

For $n \in \mathbb{N}$, we use $[n]$ to denote $\{1, \dots, n\}$. For two sets X and S, define X^S to be the set of tuples $(x_s)_{s\in S}$. We sometimes view $(x_s)_{s\in S}$ as a function from S to X. For a set S and $n \in \mathbb{N}$, $\binom{S}{n}$ denotes the collection of subsets of S of size n. Moreover, let $\binom{S}{0} = \{\emptyset\}$ and $\binom{S}{[n]} = \binom{S}{0} \cup \cdots \cup \binom{S}{n}$. For any bipartite graph (A, B, E) and $S \subseteq A, T \subseteq B$, let $E(S, T)$ denote the set of all edges with one endpoint in S and the other in T.

Let $\mathcal{X}$ be a probability distribution over a finite probability space Θ. We use $x \sim \mathcal{X}$ to denote a random variable x sampled from $\mathcal{X}$. Sometimes we abuse the notation and write Θ in place of the uniform distribution over Θ. For each $\theta \in \Theta$, we denote $\Pr_{x\sim\mathcal{X}}[x = \theta]$ by $\mathcal{X}(\theta)$. The *support* of $\mathcal{X}$ or $\mathrm{supp}(\mathcal{X})$ is the set of all $\theta \in \Theta$ such that $\mathcal{X}(\theta) \neq 0$. For any event E, we use $\mathbb{1}[E]$ to denote the indicator variable for the event.

The *informational divergence* between distributions $\mathcal{X}$ and $\mathcal{Y}$ is defined as $D_{KL}(\mathcal{X}\|\mathcal{Y}) = \sum_{\theta\in\mathrm{supp}(\mathcal{X})} \mathcal{X}(\theta)\log(\mathcal{X}(\theta)/\mathcal{Y}(\theta))$. The *total correlation* between random variables $x_1, \dots, x_n$ is $C(x_1; \dots; x_n) = D_{KL}(\mathcal{X}_{1,\dots,n}\|\mathcal{X}_1 \times \cdots \times \mathcal{X}_n)$ where $\mathcal{X}_{1,\dots,n}$ is the joint distribution of $x_1, \dots, x_n$ and $\mathcal{X}_i$ is the marginal distribution of x_i. Finally, the *conditional total correlation* is defined as $C(x_1; \dots; x_{n-1}|x_n) = \mathbb{E}_{\theta\sim\mathrm{supp}(\mathcal{X}_n)}[C(x_1; \dots; x_{n-1})|x_n = \theta]$.

For Max k-CSP, we use N to denote the instance size $(nq)^k$. For convenience, we write the predicates as $P_S(\phi|_S)$ instead of $P(S, \phi|_S)$. Moreover, for an assignment ϕ of $\mathcal{G} = (V, \mathcal{W}, \{P_S\})$, its value is $val_{\mathcal{G}}(\phi) = \mathbb{E}_{S\sim\mathcal{W}}[P_S(\phi|_S)]$. When $\mathcal{G}$ is clear from the context, we simply write $val(\phi)$. Note that $val(\mathcal{G}) = \max_\phi val_{\mathcal{G}}(\phi)$. For any $S, T \subseteq V$, $\phi_S \in \Sigma^S$ and $\phi_T \in \Sigma^T$ are said to be *consistent* if they agree on $S \cap T$ and *inconsistent* otherwise. For consistent ϕ_S, ϕ_T, we define $\phi_S \circ \phi_T \in \Sigma^{S\cup T}$ by $\phi_S \circ \phi_T(x) = \phi_S(x)$ if $x \in S$ and $\phi_S \circ \phi_T(x) = \phi_T(x)$ otherwise. Similar notations are also used for two-prover games. Finally, recall that a game $(X, Y, \mathcal{Q}, \Sigma_X, \Sigma_Y, \{P_{(x,y)}\})$ is a *projection game* if, for each $(x, y) \in \mathrm{supp}(\mathcal{Q})$, there is $f : \Sigma_X \to \Sigma_Y$ such that, for all $\sigma_x \in \Sigma_X, \sigma_y \in \Sigma_Y$, $P_{(x,y)}(\sigma_x, \sigma_y) = \mathbb{1}[f(\sigma_x) = \sigma_y]$.

3 Birthday Repetition Theorem: Proof Overview

In this section, we outline the proofs of our birthday repetition theorems. We first state our main theorems formally, starting with the birthday repetition theorem for general games.

▶ **Theorem 12.** *There exists a constant $\alpha > 0$ such that the following is true. Let $\mathcal{G} = (X, Y, E, \Sigma_X, \Sigma_Y, \{P_{(x,y)}\})$ be any two-prover game of value $1 - \varepsilon$. Let d_{max} be the maximum degree of a vertex in (X, Y, E) and $c = \log |\Sigma_X||\Sigma_Y|$. For all $0 \le k \le |X|$ and $0 \le l \le |Y|$,*

$$val(\mathcal{G}^{k\times l}) \le 2(1 - \varepsilon/2)^{\frac{\alpha\varepsilon^5 kl|E|}{d_{max}|X||Y|c^2}}$$

For projection games, we can improve the dependency on ε and avoid the dependency on c:

▶ **Theorem 13.** *There exists a constant $\alpha > 0$ such that the following is true. Let $\mathcal{G} = (X, Y, E, \Sigma_X, \Sigma_Y, \{P_{(x,y)}\})$ be any projection game of value $1 - \varepsilon$. Let d_{max} be the maximum degree of a vertex in (X, Y, E). For all $0 \le k \le |X|$ and $0 \le l \le |Y|$, we have*

$$val(\mathcal{G}^{k\times l}) \le 2(1 - \varepsilon/2)^{\frac{\alpha\varepsilon^3 kl|E|}{d_{max}|X||Y|}}$$

In short, we will to show that $\mathcal{G}^{k\times l}$ has small value by "embedding" an $\Omega\left(\frac{kl|E|}{d_{max}|X||Y|}\right)$-parallel repetition game, which has low value by the parallel repetition theorem, into it.

For convenience, let s denote $\frac{kl|E|}{|X||Y|}$, the expected number of edges in $E(S, T)$ when S and T are independently uniformly sampled from $\binom{X}{k}$ and $\binom{Y}{l}$ respectively. Let s_1 and s_2 be $s(1 + \delta)$ and $s(1 - \delta)$ respectively for some $\delta \in [0, 1/2]$ that will be chosen later. We will use $r = \beta s/d_{max}$ rounds of parallel repetition where $\beta \in [0, \delta/40]$ will be specified later. Lastly, let $E^r = \{((x_1, \ldots, x_r), (y_1, \ldots, y_r)) \mid (x_1, y_1), \ldots, (x_r, y_r) \in E\}$.

▶ Remark. δ and β will be chosen based on ε, c and whether $\mathcal{G}$ is a projection game. When ε and c are constant, both δ and β are small constants. This is the most representative case and is good to keep in mind when reading through the proof.

Our overall strategy is to reduce $\mathcal{G}^{\otimes r}$ to $\mathcal{G}^{k\times l}$. Since $val(\mathcal{G}^{\otimes r})$ is exponentially small in $r = \Omega\left(\frac{kl|E|}{d_{max}|X||Y|}\right)$ due to the parallel repetition theorem, such reduction would give a similar upper bound on $val(\mathcal{G}^{k\times l})$. Unfortunately, we do not know how to do this in one step so we will have to go through a sequence of reductions. The sequence of games that we reduce to are $\mathcal{G}^{\otimes r}_{\text{set}}, \mathcal{G}^{k\times l}_{\text{em}}, \mathcal{G}^{k\times l}_{\text{em},[s_1,s_2]}$ and $\mathcal{G}^{k\times l}_{[s_1,s_2]}$ respectively. The game $\mathcal{G}^{\otimes r}_{\text{set}}$ share the same questions, alphabet sets and predicates with $\mathcal{G}^{\otimes r}$ while $\mathcal{G}^{k\times l}_{\text{em}}, \mathcal{G}^{k\times l}_{\text{em},[s_1,s_2]}$ and $\mathcal{G}^{k\times l}_{[s_1,s_2]}$ share those with $\mathcal{G}^{k\times l}$. The distribution of each game is defined as follows.

- The distribution of $\mathcal{G}^{\otimes r}_{\text{set}}$ is uniform over the set E^r_{set} of all $((x_1, \ldots, x_r), (y_1, \ldots, y_r)) \in E^r$ such that $x_1, \ldots, x_r, y_1, \ldots, y_r$ are all distinct. Note that this distribution is simply $\mathcal{G}^{\otimes r}$'s distribution conditioned on $x_1, \ldots, x_r, y_1, \ldots, y_r$ being all distinct.
- We will try to make the distribution $\mathcal{Q}^{k\times l}_{\text{em}}$ of $\mathcal{G}^{k\times l}_{\text{em}}$ reflect an embedding of the game $\mathcal{G}^{\otimes r}_{\text{set}}$. We define $\mathcal{Q}^{k\times l}_{\text{em}}$ based on the following sampling process for $(S, T) \sim \mathcal{Q}^{k\times l}_{\text{em}}$. First, sample $((x_1, \ldots, x_r), (y_1, \ldots, y_r))$ uniformly at random from E^r_{set}. Then, sample $\tilde{S}$ and $\tilde{T}$ independently uniformly from $\binom{X-\{x_1,\ldots,x_r\}}{k-r}$ and $\binom{Y-\{y_1,\ldots,y_r\}}{l-r}$ respectively. Finally, set $S = \{x_1, \ldots, x_r\} \cup \tilde{S}$ and $T = \{y_1, \ldots, y_r\} \cup \tilde{T}$.
- The distribution $\mathcal{Q}^{k\times l}_{\text{em},[s_1,s_2]}$ of $\mathcal{G}^{k\times l}_{\text{em},[s_1,s_2]}$ is the distribution $\mathcal{Q}^{k\times l}_{\text{em}}$ conditioned on the number of edges between the two sets being in the range $[s_1, s_2]$. In other words, $\mathcal{Q}^{k\times l}_{\text{em},[s_1,s_2]}(S, T) = \Pr_{(S',T')\sim\mathcal{Q}^{k\times l}_{\text{em}}}[S = S' \wedge T = T' \mid s_1 \le |E(S', T')| \le s_2]$.

- Finally, the distribution of $\mathcal{G}^{k\times l}_{[s_1,s_2]}$ is uniform over the set $E^{k\times l}_{[s_1,s_2]}$ of all (S,T) such that $|E(S,T)| \in [s_1,s_2]$. In other words, we ignore weights in $\mathcal{Q}^{k\times l}_{\mathrm{em},[s_1,s_2]}$ and use the uniform distribution over $\mathrm{supp}(\mathcal{Q}^{k\times l}_{\mathrm{em},[s_1,s_2]})$.

Before we present the overview of the proofs, let us list simple bounds that will be useful in understanding the intuitions. Their proofs can be found in the full version of this work [42].

▶ **Lemma 14.** *Let (X,Y,E) be any bipartite graph with maximum degree d_{max}. For any non-negative integers $k \le |X|$ and $l \le |Y|$, let $s = \frac{kl|E|}{|X||Y|}$. For any $0 \le \gamma < 1/2$, we have*

$$\Pr_{S\sim\binom{X}{k},T\sim\binom{Y}{l}}[|E(S,T)| \notin [(1-\gamma)s,(1+\gamma)s]] \le 4\exp\left(-\frac{\gamma^2 s}{54 d_{max}}\right).$$

▶ **Lemma 15.** *Let $\mathcal{G}$ and $\mathcal{G}'$ be two games on the same questions, alphabets, and predicates but on different distributions $\mathcal{Q}$ and $\mathcal{Q}'$ respectively. If, for some α, $\mathcal{Q}(x,y) \le \alpha\cdot\mathcal{Q}'(x,y)$ for all $x\in X, y\in Y$, then $val(\mathcal{G}) \le \alpha\cdot val(\mathcal{G}')$. In particular, when $\mathcal{Q}$ and $\mathcal{Q}'$ are uniform distributions on some $E\subseteq E'$, $val(\mathcal{G}) \le \frac{|E'|}{|E|}\cdot val(\mathcal{G}')$.*

▶ **Lemma 16.** *Let $\mathcal{G} = (X,Y,\mathcal{Q},\Sigma_X,\Sigma_Y,\{P_{x,y}\}_{(x,y)\in\mathrm{supp}(\mathcal{Q})})$ be any two player game and let A be any event occurring with probability $1-p>0$ (w.r.t. $\mathcal{Q}$). Let $\mathcal{Q}'$ be the conditional probability $\mathcal{Q}$ given A, i.e., $\mathcal{Q}'(\tilde{x},\tilde{y}) = \Pr_{(x,y)\sim\mathcal{Q}}[x=\tilde{x}\wedge y=\tilde{y} \mid A]$. For the game $\mathcal{G}' = (X,Y,\mathcal{Q}',\Sigma_X,\Sigma_Y,\{P_{x,y}\}_{(x,y)\in\mathrm{supp}(\mathcal{Q}')})$, we have $val(\mathcal{G}) - p \le val(\mathcal{G}') \le val(\mathcal{G}) + 2p$.*

We will next give intuitions on why $val(\mathcal{G}^{\otimes r}) \approx val(\mathcal{G}^{\otimes r}_{\mathrm{set}}) \approx val(\mathcal{G}^{k\times l}_{\mathrm{em}}) \approx val(\mathcal{G}^{k\times l}_{\mathrm{em},[s_1,s_2]}) \approx val(\mathcal{G}^{k\times l}_{[s_1,s_2]}) \approx val(\mathcal{G}^{k\times l})$ where each $\approx$ hides some multiplicative or additive losses in each step. With the right choice of δ and β, we can ensure that each loss is significantly smaller than $val(\mathcal{G}^{\otimes r})$, and, thus, we will be able to bound $val(\mathcal{G}^{k\times l})$. Below, we state these losses more precisely and summarize the overview of each proof.

▶ **Lemma 17.** $val(\mathcal{G}^{\otimes r}_{set}) \le \left(\frac{1}{1-2\beta}\right)^r \cdot val(\mathcal{G}^{\otimes r})$

Proof Idea. From Lemma 15, it suffices to lower bound the ratio $|E^r_{\mathrm{set}}|/|E^r|$. This is the probability that r random edges from E do not share any endpoints, which is easy to bound. ◀

▶ **Lemma 18.** $val(\mathcal{G}^{k\times l}_{em}) \le val(\mathcal{G}^{\otimes r}_{set})$

Proof Idea. Based on how $\mathcal{Q}^{k\times l}_{\mathrm{em}}$ is defined, it induces a canonical map from each strategy in $\mathcal{G}^{k\times l}_{\mathrm{em}}$ to a "mixed strategy" in $\mathcal{G}^{\otimes r}_{\mathrm{set}}$. We can show that each strategy ϕ in $\mathcal{G}^{k\times l}_{\mathrm{em}}$ has value no more than the value of the mixed strategy in $\mathcal{G}^{\otimes r}_{\mathrm{set}}$ that ϕ maps to, which yields the lemma. ◀

▶ **Lemma 19.** $val(\mathcal{G}^{k\times l}_{em,[s_1,s_2]}) \le val(\mathcal{G}^{k\times l}_{em}) + 8\exp\left(-\frac{\delta^2 r}{432\beta}\right)$

Proof Idea. $\mathcal{Q}^{k\times l}_{\mathrm{em},[s_1,s_2]}$ is $\mathcal{Q}^{k\times l}_{\mathrm{em}}$ conditioned on the event $E(S,T)\in[s_1,s_2]$. From Lemma 16, it suffices to bound the probability of such event. From the definition of $\mathcal{Q}^{k\times l}_{\mathrm{em}}$, S and T can be sampled by first sampling $x_1,\ldots,x_r,y_1,\ldots,y_r$ according to E^r and then sampling the rest of S and T from $X-\{x_1,\ldots,x_r\}$ and $Y-\{y_1,\ldots,y_r\}$ respectively. When r is small enough, we can show, with the help of Lemma 14, that, for any $x_1,\ldots,x_r,y_1,\ldots,y_r$, $|E(S,T)|$ concentrates around s. This gives us the desired bound. ◀

▶ **Lemma 20.** $val(\mathcal{G}^{k\times l}_{[s_1,s_2]}) \le \left(\frac{1+\delta}{1-\delta-2\beta}\right)^{2r} \cdot val(\mathcal{G}^{k\times l}_{em,[s_1,s_2]})$

Proof Idea. We will show that the two distributions are (multiplicatively) close and evoke Lemma 15 to arrive at the bound. Since the distribution of $\mathcal{G}^{k\times l}_{[s_1,s_2]}$ is uniform, we only need to show that the maximum and the minimum (non-zero) probabilities in $\mathcal{Q}^{k\times l}_{\text{em},[s_1,s_2]}$ are close.

Fortunately, we know that $\mathcal{Q}^{k\times l}_{\text{em},[s_1,s_2]}$ is $\mathcal{Q}^{k\times l}_{\text{em}}$ conditioned on an event. This means that, when $\mathcal{Q}^{k\times l}_{\text{em},[s_1,s_2]}(S,T)$ is not zero, it is proportional to $\mathcal{Q}^{k\times l}_{\text{em}}(S,T)$. The latter, in turn, is proportional to the number of edges $(x_1,y_1),\ldots,(x_r,y_r)\in E^r$ such that $x_1,\ldots,x_r,y_1,\ldots,y_r$ are all distinct and $x_1,\ldots,x_r\in S$ and $y_1,\ldots,y_r\in T$. In other words, we want to upper bound and lower bound the number of r edges in $E(S,T)$ with distinct endpoints. This is feasible since we know that $|E(S,T)|\in[s_1,s_2]$ and r is so small that with a reasonable probability r edges picked will not share any endpoint with each other. ◀

▶ **Lemma 21.** $val(\mathcal{G}^{k\times l})\le val(\mathcal{G}^{k\times l}_{[s_1,s_2]})+4\exp\left(-\frac{\delta^2 r}{54\beta}\right)$

Proof Idea. By realising that $\mathcal{G}^{k\times l}_{[s_1,s_2]}$'s distribution is simply $\mathcal{G}^{k\times l}$'s distribution conditioned on $|E(S,T)|\in[s_1,s_2]$, this follows immediately from Lemma 14 and Lemma 16. ◀

We defer proofs of the above lemmas to the full version [42]. Let us now use them to prove the birthday repetition theorems. To avoid repeating arguments for general games and projection games, we prove the following lemma. Its proof, mostly calculations, is deferred to the full version.

▶ **Lemma 22.** *Let $\mathcal{G}$ be any game of value $1-\varepsilon$ and k,l,β,δ,r be as above. If $val(\mathcal{G}^{\otimes r})\le(1-\varepsilon/2)^R$ for some R such that $\frac{200\delta r}{\varepsilon}\le R\le\min\{r,\frac{\delta^2 r}{1000\beta\varepsilon}\}$, then $val(\mathcal{G}^{k\times l})\le 2(1-\varepsilon/2)^{R/10}$.*

The final ingredient for our main proof is the parallel repetition theorem. For general games, we use Holenstein's version of the theorem [34], which is stated below.

▶ **Theorem 23** ([34])**.** *There is a constant $C>0$ such that, for every $k>0$ and any two-prover game $\mathcal{G}=(X,Y,\mathcal{Q},\Sigma_X,\Sigma_Y,\{P_{(x,y)}\})$ of value $1-\varepsilon$, $val(\mathcal{G}^{\otimes k})\le(1-\varepsilon/2)^{C\varepsilon^2 k/\log(|\Sigma_X||\Sigma_Y|)}$.*

Equipped with Lemma 22 and the parallel repetition theorem, we can now prove our birthday repetition theorems just by selecting the right δ and β.

Proof of Theorem 12. Pick $\delta=\frac{\varepsilon^3 C}{10^3 c}$ and $\beta=\frac{\varepsilon^3 C}{10^{10}c}$ where C is the constant from Theorem 23. From Theorem 23, we have $val(\mathcal{G}^{\otimes r})\le(1-\varepsilon/2)^{C\varepsilon^2 r/c}$. Let $R=C\varepsilon^2 r/c$. We can see that R,δ,β satisfy the conditions in Lemma 22. Hence, we can conclude that $val(\mathcal{G}^{k\times l})\le 2(1-\varepsilon/2)^{R/10}=2(1-\varepsilon/2)^{(C^2/10^{11})\left(\frac{\varepsilon^5 kl|E|}{c^2|X||Y|d_{max}}\right)}$ as desired. ◀

In the case of projection games, we can improve dependency on ε and get rid of dependency on c thanks to the stronger bound in Rao's parallel repetition theorem for projection games [50].

▶ **Theorem 24** ([50])**.** *There exists a constant $C>0$ such that, for any projection game $\mathcal{G}$ of value $1-\varepsilon$ and for every $k>0$, we have $val(\mathcal{G}^{\otimes k})\le(1-\varepsilon/2)^{C\varepsilon k}$.*

Proof of Theorem 13. Pick $\delta=\frac{\varepsilon^2 C}{10^3}$ and $\beta=\frac{\varepsilon^2 C}{10^{10}}$ where C is the constant from Theorem 24. From the theorem, we have $val(\mathcal{G}^{\otimes r})\le(1-\varepsilon/2)^{C\varepsilon r}$. Let $R=C\varepsilon r$. By evoking Lemma 22, we have $val(\mathcal{G}^{k\times l})\le 2(1-\varepsilon/2)^{R/10}=2(1-\varepsilon/2)^{(C^2/10^{11})\left(\frac{\varepsilon^3 kl|E|}{|X||Y|d_{max}}\right)}$ as desired. ◀

4 Improved Approximation Algorithm for Dense CSPs

To describe our algorithm, we first explain ingredients central in conditioning-based algorithms: a LP/SDP hierarchy, a conditioning operator, and an independent rounding procedure.

Sherali-Adams (SA) relaxation of Max k-CSP. An *r-level SA solution* μ of $\mathcal{G} = (V, \mathcal{W}, \{P_S\})$ is a collection $\{\mathcal{X}_S\}$ of distributions $\mathcal{X}_S$ on Σ^S for every $S \in \binom{V}{[r]}$ such that, for every $S, T \in \binom{V}{[r]}$, the marginal probability of $\mathcal{X}_S$ and $\mathcal{X}_T$ on $\Sigma^{S\cap T}$ agrees. For $r \geq k$, the value of μ is $val_{SA}(\mu) = \mathbb{E}_{S\sim\mathcal{W}}[\mathbb{E}_{x_S\sim\mu}[P_S(x_S)]]$ where $\mathbb{E}_{x_S\sim\mu}[P_S(x_S)]$ is a shorthand for $\mathbb{E}_{\phi_S\sim\mathcal{X}_{\{i_1,\dots,i_k\}}}[P_S(\phi_S)]$ when $S = (x_{i_1}, \dots, x_{i_k})$. The optimum of the r-level SA relaxation of $\mathcal{G}$, $opt^r_{SA}(\mathcal{G})$, is the maximum value among all the r-level SA solutions. Clearly, finding $opt^r_{SA}(\mathcal{G})$ can be formulated as a LP and can be computed in $(nq)^{O(r)}$ time.

Conditioning SA Solution. Let $\mu = \{\mathcal{X}_S\}$ be any r-level SA solution. For any $T \subseteq V$ and $\phi_T \in \Sigma^T$ such that $\mathcal{X}_T(\phi_T) > 0$, μ conditioned on ϕ_T is $\mu|\phi_T = \{\tilde{\mathcal{X}}_S\}_{|S|\leq r-|T|}$ where

$$\tilde{\mathcal{X}}_S(\phi_S) = \begin{cases} \mathcal{X}_{S\cup T}(\phi_S \circ \phi_T)/\mathcal{X}_T(\phi_T) & \text{if } \phi_S \text{ is consistent with } \phi_T, \\ 0 & \text{otherwise.} \end{cases}$$

It is not hard to see that $\mu|\phi_T$ is an $(r - |T|)$-level SA solution.

Independent Rounding. A natural way to round a SA solution $\{\mathcal{X}_S\}$ is to independently assign each variable x based on $\mathcal{X}_x$. This gives a solution with expected value at least $\mathbb{E}_{S=(x_{i_1},\dots,x_{i_k})\sim\mathcal{W}}\left[\mathbb{E}_{\phi_S\sim\mathcal{X}_{i_1}\times\cdots\times\mathcal{X}_{i_k}}[P_S(\phi_S)]\right]$ and can be easily derandomized.

Without going into too much detail, conditioning-based algorithms typically proceed as follows. First, solve a LP/SDP relaxation of the problem. As long as the solution has large "total correlation", try conditioning it on an assignment to a random variable. Once the solution has small total correlation, use independent rounding on the solution to get the desired assignment. The intuition here is that, if the solution has large total correlation, conditioning on one variable substantially reduces the total correlation. Hence, after a certain number of rounds of conditioning, the total correlation becomes small. At this point, the solution is quite independent and independent rounding gives a good approximation.

Our algorithm will also follow this framework. In fact, it remains largely unchanged from [60] except that we use a stronger relaxation to avoid arguing about values of conditioned solutions. However, our main contribution lies in the analysis: we will show that independent rounding does well even when the total correlation is large (super-constant). This is in contrast to the previously known conditioning-based algorithms [11, 49, 60], all of which require their measures of correlation to be small constants to get any meaningful result.

The new relaxation, which we call the r-level SA with Conditioning (SAC), is defined below.

$$\begin{aligned} &\text{maximize } \lambda \\ &\text{subject to } \{\mathcal{X}_S\}_{|S|\leq r} \text{ is a valid } r\text{-level SA solution} \\ &\qquad \mathop{\mathbb{E}}_{S\sim\mathcal{W}}\Big[\mathop{\mathbb{E}}_{\phi_S\sim(\mu|\phi_T)}[P_S(\phi_S)]\Big] \geq \lambda \qquad\qquad \forall T, \phi_T \text{ s.t. } |T| \leq r-k, \mathcal{X}_T(\phi_T) > 0. \end{aligned}$$

If λ is a constant, the program can be easily written as a LP. Thus, the relaxation can be solved to within arbitrarily small error in $(nq)^{O(r)}$ time by binary search on λ.

Algorithm 1 Approximation Algorithm for Dense CSPs

Input: a Δ-dense MAX k-CSP instance $\mathcal{G}$, an integer i
Output: An assignment $\phi : V \to \Sigma$

```
r ← k, λ ← 0
while (r − k)λ < k²i/Δ and r < n do
    r ← r + 1
    μ, λ ← solve r-level of SAC relaxation for G
for T ∈ (V choose [r−k]), φ_T ∈ Σ^T do
    φ ← independent rounding of μ|φ_T
return φ from the previous step with maximum value
```

Figure 1 Approximation Algorithm for Dense CSPs. The difference between this and the above summary is that we iteratively increase the number of levels r. This is because the number of levels depends on the value of the solution (see Lemma 28). Specifically, we need $r \geq k^2 i/(\Delta\lambda) + k$.

Roughly speaking, our algorithm first solves an $O(\frac{k^2 i}{\Delta} + k)$-level SAC relaxation for the instance. We then try every possible conditioning (i.e., every assignment to $T \subseteq V$ of size $\leq k^2 i/\Delta$). For each conditioned solution, we use independent rounding to arrive at an assignment. Finally, output the best such assignment. The pseudo-code for the full algorithm is shown in Figure 1. This algorithm yields the following approximation for the problem.

▶ **Theorem 25** (Theorem 8, Restated). *On any Δ-dense* MAX k-CSP *instance of value $1-\delta$, Algorithm 1 outputs an assignment of value at least $(1-\delta)\delta^{\frac{\delta}{1-\delta}}/q^{1/i}$ in time $N^{O\left(\frac{ki}{(1-\delta)\Delta}\right)}$.*

We spend the rest of the section sketching the proof of Theorem 25. First, we define and state a bound on the total correlation of conditioned solutions in Section 4.1. Then, in Subsection 4.2, we state our main contribution of this section, i.e., that even when the total correlation is super-constant, independent rouding still yields non-trivial approximation.

4.1 Total Correlation of Conditioned Sherali-Adams Solution

For a k-level SA solution $\mu = \{\mathcal{X}_S\}$ and a tuple $S = (x_{i_1}, \ldots, x_{i_j}) \in V^j$ where $j \leq k$, the total correlation of S is $C_\mu(x_S) = C(\sigma_{i_1}; \ldots; \sigma_{i_j})$ where $\sigma_{i_1}, \ldots, \sigma_{i_j}$ are jointly sampled from $\mathcal{X}_{\{x_{i_1}, \ldots, x_{i_j}\}}$. The total correlation of μ is then defined as $C(\mu) = \mathbb{E}_{S\sim\mathcal{W}}[C_\mu(x_S)]$. μ is said to be *κ-independent* if $C(\mu) \leq \kappa$. Yoshida and Zhou [60] show that, for any $l > 0$ and any $(l+k)$-level SA solution μ, there exists an assignment $\phi_T \in \Sigma^T$ to a subset T of size $\leq l$ such that the total correlation of $(\mu|\phi_T)$ is at most $3^k \log q/(l\Delta)$. Here we can improve this bound as stated below. Since the proof is similar to that of [60], we defer it to the full version of this work [42].

▶ **Lemma 26.** *Let μ be a r-level SA solution of a Δ-dense* MAX k-CSP *instance $(V, \mathcal{W}, \{P_S\})$. Then, for any $0 < l \leq r - k$, there is $t \leq l$ such that $\mathbb{E}_{T\sim V^t, \phi_T\sim\Sigma^T}[C(\mu|\phi_T)] \leq \frac{k^2 \log q}{l\Delta}$.*

4.2 New Bound on Rounding κ-independent Solution

For the known conditioning-based algorithms, once the solution is fairly independent, it is easy to show that independent rounding gives a good solution. Specifically, [49] and [60] conclude this step using the Pinsker's inequality, which states that, for any distributions $\mathcal{X}$ and $\mathcal{Y}$, $D_{KL}(\mathcal{X}\|\mathcal{Y}) \geq (2\log 2)\|\mathcal{X} - \mathcal{Y}\|_1^2$. Roughly speaking, $\mathcal{X}$ is the distribution in the LP

solution whereas $\mathcal{Y}$ is the distribution from independent rounding. Hence, once $D_{KL}(\mathcal{X}\|\mathcal{Y})$ is at most a small constant ε, it follows that, for any predicate f, $|\mathbb{E}_{x\sim\mathcal{X}}[f(x)] - \mathbb{E}_{y\sim\mathcal{Y}}[f(y)]| \leq \sqrt{\varepsilon/(2\log 2)}$. Thus, if $\mathbb{E}_{x\sim\mathcal{X}}[f(x)]$, the value of the LP solution, is large, then $\mathbb{E}_{y\sim\mathcal{Y}}[f(y)]$, the expected value of a solution from independent rouding, is also large.

While this works for small ε, it completely fails when ε is larger than a certain constant. In this regard, we prove the following lemma, which gives a non-trivial bound even for large ε. For convenience, 0^0 is defined to be 1 and $(\delta^\delta e^{-\kappa})^{\frac{1}{1-\delta}}(1-\delta)$ is defined to be 0 when $\delta = 1$.

▶ **Lemma 27.** *For any two probability distributions $\mathcal{X}, \mathcal{Y}$ over Θ such that $D_{KL}(\mathcal{X}\|\mathcal{Y}) \leq \kappa$ and any $f : \Theta \to [0,1]$, if $\mathbb{E}_{x\sim\mathcal{X}}[f(x)] = 1-\delta$, then $\mathbb{E}_{y\sim\mathcal{Y}}[f(y)] \geq \left(\delta^\delta e^{-\kappa}\right)^{\frac{1}{1-\delta}}(1-\delta)$.*

Lemma 27 can then be used to prove a new lower bound for the value of the output from independent rounding on a κ-independent k-level SA solution as stated below.

▶ **Lemma 28.** *If $\{\mathcal{X}_S\}$ is a κ-independent k-level SA solution of value $1-\delta$ for a* Max k-CSP *instance, then independent rounding gives an assignment of value at least $(\delta^\delta e^{-\kappa})^{\frac{1}{1-\delta}}(1-\delta)$.*

Theorem 25 can now be proved by combining Lemma 26 and 28. Due to space constraint, we omit the proofs of Lemma 27, Lemma 28 and Theorem 25 from this extended abstract.

5 Conclusion and Open Problems

While we settle down the approximability of dense Max k-CSP up to a $k\,\mathrm{polylog}(ki)$ factor in the exponent, our work raises many interesting questions such as the two listed below:

- *Can Lemma 27 be used to prove new approximation guarantees for other problems?* Lemma 27 is a generic bound relating expectations of a function on two distributions based on their informational divergence. Thus, it may help yield new approximation guarantees for other correlation-based algorithms.
- *What is the right dependency on ε and c in the birthday repetition theorem?* It is likely that the dependency of ε and c in our birthday repetition is not tight. In particular, parallel repetition for general games only has $1/c$ factor in the exponent whereas our theorem has $1/c^2$; would it be possible to reduce the dependency to $1/c$ in birthday repetition? Similar question also applies to ε.

Acknowledgements. PM would like to thank Aviad Rubinstein, Dana Moshkovitz, Grigory Yaroslavtsev and Madhur Tulsiani for useful discussions. We also thank Irit Dinur for informing us about her result based on gap-ETH.

References

1 Scott Aaronson, Salman Beigi, Andrew Drucker, Bill Fefferman, and Peter W. Shor. The power of unentanglement. *Theory of Computing*, 5(1):1–42, 2009.
2 Scott Aaronson, Russell Impagliazzo, and Dana Moshkovitz. AM with multiple Merlins. In *IEEE CCC*, pages 44–55, June 2014.
3 Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of MAX-CSPs. *J. Comput. Syst. Sci.*, 67(2):212–243, September 2003.
4 Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.*, 42(5):2008–2037, 2013.
5 Sanjeev Arora, David Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *ACM STOC*, pages 284–293, 1995.

6 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

7 Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

8 Yakov Babichenko, Christos H. Papadimitriou, and Aviad Rubinstein. Can almost everybody be almost happy? In *ACM ITCS*, pages 1–9, 2016.

9 Boaz Barak, Fernando G. S. L. Brandao, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *ACM STOC*, pages 307–326, 2012.

10 Boaz Barak, Moritz Hardt, Thomas Holenstein, and David Steurer. Subsampling mathematical relaxations and average-case complexity. In *ACM-SIAM SODA*, pages 512–531, 2011.

11 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *IEEE FOCS*, pages 472–481, 2011.

12 Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russell. Efficient probabilistically checkable proofs and applications to approximations. In *ACM STOC*, pages 294–304, 1993.

13 Aditya Bhaskara, Moses Charikar, Eden Chlamtác, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *ACM STOC*, pages 201–210, 2010.

14 Mark Braverman and Ankit Garg. Small value parallel repetition for general games. In *ACM STOC*, pages 335–340, 2015.

15 Mark Braverman, Young Kun Ko, Aviad Rubinstein, and Omri Weinstein. ETH hardness for densest-k-subgraph with perfect completeness. In *ACM-SIAM SODA*, pages 1326–1341, 2017.

16 Mark Braverman, Young Kun Ko, and Omri Weinstein. Approximating the best Nash equilibrium in $n^{o(\log n)}$-time breaks the exponential time hypothesis. In *ACM-SIAM SODA*, pages 970–982, 2015.

17 Moses Charikar, MohammadTaghi Hajiaghayi, and Howard Karloff. Improved approximation algorithms for label cover problems. *Algorithmica*, 61(1):190–206, 2011.

18 Eden Chlamtác, Michael Dinitz, Christian Konrad, Guy Kortsarz, and George Rabanca. The densest k-subhypergraph problem. In *APPROX*, pages 6:1–6:19, 2016.

19 Eden Chlamtác, Pasin Manurangsi, Dana Moshkovitz, and Aravindan Vijayaraghavan. Approximation algorithms for label cover and the log-density threshold. In *ACM-SIAM SODA*, pages 900–919, 2017.

20 Amin Coja-Oghlan, Colin Cooper, and Alan Frieze. An efficient sparse regularity concept. *SIAM J. Discret. Math.*, 23(4):2000–2034, 2010.

21 Wenceslas Fernandez de la Vega, Marek Karpinski, Ravi Kannan, and Santosh Vempala. Tensor decomposition and approximation schemes for constraint satisfaction problems. In *ACM STOC*, pages 747–754, 2005.

22 Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. Linear programming relaxations of Maxcut. In *ACM-SIAM SODA*, pages 53–61, 2007.

23 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *ECCC*, 23:128, 2016.

24 Irit Dinur, Eldar Fischer, Guy Kindler, Ran Raz, and Shmuel Safra. PCP characterizations of NP: Toward a polynomially-small error-probability. *Computational Complexity*, 20(3):413–504, 2011.

25 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *ACM STOC*, pages 624–633, 2014.

26 Shaddin Dughmi. On the hardness of signaling. In *IEEE FOCS*, pages 354–363, 2014.

27 Uriel Feige, David Peleg, and Guy Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3), 2001.

28 Uriel Feige and Michael Seltser. On the densest k-subgraph problems, 1997.

29 Dimitris Fotakis, Michael Lampis, and Vangelis Th. Paschos. Sub-exponential approximation schemes for CSPs: From dense to almost sparse. In *STACS*, pages 37:1–37:14, 2016.

30 Alan M. Frieze and Ravi Kannan. The regularity lemma and approximation schemes for dense problems. In *FOCS*, pages 12–20, 1996.

31 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives. In *IEEE FOCS*, pages 482–491, 2011.

32 Mohammad Taghi Hajiaghayi, Kamal Jain, Kishori M. Konwar, Lap Chi Lau, Ion I. Măndoiu, Alexander Russell, Alexander A. Shvartsman, and Vijay V. Vazirani. The minimum k-colored subgraph problem in haplotyping and DNA primer selection. In *IWBRA*, 2006.

33 Aram W. Harrow and Ashley Montanaro. Testing product states, quantum Merlin-Arthur games and tensor optimization. *J. ACM*, 60(1):3:1–3:43, February 2013.

34 Thomas Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(1):141–172, 2009.

35 Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, March 2001.

36 Guy Kortsarz and David Peleg. On choosing a dense subgraph. In *IEEE SFCS*, pages 692–701, 1993.

37 Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. on Optimization*, 11(3):796–817, March 2000.

38 Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *ACM EC*, pages 36–41, 2003.

39 Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *ACM STOC*, 2017. To appear.

40 Pasin Manurangsi and Dana Moshkovitz. Approximating dense Max 2-CSPs. In *APPROX*, pages 396–415, 2015.

41 Pasin Manurangsi and Dana Moshkovitz. Improved approximation algorithms for projection games. *Algorithmica*, pages 1–40, 2015.

42 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense CSPs. *CoRR*, abs/1607.02986, 2016. URL: `https://arxiv.org/abs/1607.02986`.

43 Pasin Manurangsi and Aviad Rubinstein. Inapproximability of VC Dimension and Littlestone's Dimension. Unpublished manuscript, 2017.

44 Claire Mathieu and Warren Schudy. Yet another algorithm for dense max cut: go greedy. In *ACM-SIAM SODA*, pages 176–182, 2008.

45 Dana Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. In *APPROX 2012*, pages 276–287, 2012.

46 Dana Moshkovitz. Parallel repetition from fortification. In *IEEE FOCS*, pages 414–423, 2014.

47 Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5):29:1–29:29, June 2008.

48 David Peleg. Approximation algorithms for the label-cover max and red-blue set cover problems. *Journal of Discrete Algorithms*, 5(1):55–64, 2007.

49 Prasad Raghavendra and Ning Tan. Approximating CSPs with global cardinality constraints using SDP hierarchies. In *ACM-SIAM SODA*, pages 373–387, 2012.

50 Anup Rao. Parallel repetition in projection games and a concentration bound. *SIAM J. Comput.*, 40(6):1871–1891, 2011.

51 Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.

52 Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *ACM STOC*, pages 475–484, 1997.

53 Aviad Rubinstein. ETH-hardness for signaling in symmetric zero-sum games. *CoRR*, abs/1510.04991, 2015.

54 Aviad Rubinstein. Settling the complexity of computing approximate two-player Nash equilibria. In *IEEE FOCS*, pages 258–265, 2016.

55 Grant Schoenebeck. Linear level lasserre lower bounds for certain k-CSPs. In *IEEE FOCS*, pages 593–602, 2008.

56 Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxation between the continuous and convex hull representations. *SIAM J. Discret. Math.*, 3(3):411–430, May 1990.

57 Akiko Suzuki and Takeshi Tokuyama. Dense subgraph problems with output-density conditions. *ACM Trans. Algorithms*, 4(4):43:1–43:18, August 2008.

58 Grigory Yaroslavtsev. Going for speed: Sublinear algorithms for dense r-CSPs. *CoRR*, abs/1407.7887, 2014.

59 Grigory Yaroslavtsev. Personal communication, March 2016.

60 Yuichi Yoshida and Yuan Zhou. Approximation schemes via Sherali-Adams hierarchy for dense constraint satisfaction problems and assignment problems. In *ITCS 2014*, pages 423–438, 2014.

Inapproximability of Maximum Edge Biclique, Maximum Balanced Biclique and Minimum k-Cut from the Small Set Expansion Hypothesis*†

Pasin Manurangsi

University of California, Berkeley, CA, USA
pasin@berkeley.edu

Abstract

The *Small Set Expansion Hypothesis* (SSEH) is a conjecture which roughly states that it is NP-hard to distinguish between a graph with a small set of vertices whose expansion is almost zero and one in which all small sets of vertices have expansion almost one. In this work, we prove conditional inapproximability results for the following graph problems based on this hypothesis:

- *Maximum Edge Biclique* (MEB): given a bipartite graph G, find a complete bipartite subgraph of G with maximum number of edges. We show that, assuming SSEH and that NP $\not\subseteq$ BPP, no polynomial time algorithm gives $n^{1-\varepsilon}$-approximation for MEB for every constant $\varepsilon > 0$.
- *Maximum Balanced Biclique* (MBB): given a bipartite graph G, find a balanced complete bipartite subgraph of G with maximum number of vertices. Similar to MEB, we prove $n^{1-\varepsilon}$ ratio inapproximability for MBB for every $\varepsilon > 0$, assuming SSEH and that NP $\not\subseteq$ BPP.
- *Minimum k-Cut*: given a weighted graph G, find a set of edges with minimum total weight whose removal splits the graph into k components. We prove that this problem is NP-hard to approximate to within $(2 - \varepsilon)$ factor of the optimum for every $\varepsilon > 0$, assuming SSEH.

The ratios in our results are essentially tight since trivial algorithms give n-approximation to both MEB and MBB and 2-approximation algorithms are known for Minimum k-Cut [35].

Our first two results are proved by combining a technique developed by Raghavendra, Steurer and Tulsiani [33] to avoid locality of gadget reductions with a generalization of Bansal and Khot's long code test [4] whereas our last result is shown via an elementary reduction.

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

Keywords and phrases Hardness of Approximation, Small Set Expansion Hypothesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.79

1 Introduction

Since the PCP theorem was proved two decades ago [2, 3], our understanding of approximability of combinatorial optimization problems has grown enormously; tight inapproximability results have been obtained for fundamental problems such as Max-3SAT [15], Max Clique [14] and Set Cover [27, 9]. Yet, for other problems, including Vertex Cover and Max Cut, known NP-hardness of approximation results come short of matching best known algorithms.

The introduction of the Unique Games Conjecture (UGC) by Khot [19] propelled another wave of development in hardness of approximation that saw many of these open problems resolved (see e.g. [23, 21]). Alas, some problems continue to elude even attempts at proving

* A full version of the paper is available at https://arxiv.org/abs/1705.03581.

† This material is based upon work supported by the National Science Foundation under Grants No. CCF 1540685 and CCF 1655215.

UGC-hardness of approximation. For a class of such problems, the failure stems from the fact that typical reductions are local in nature; many reductions from unique games to graph problems could produce disconnected graphs. If we try to use such reductions for problems that involve some forms of expansion of graphs (e.g. Sparsest Cut), we are out of luck.

One approach to overcome the aforementioned issue is through the *Small Set Expansion Hypothesis* (SSEH) of Raghavendra and Steurer [32]. To describe the hypothesis, recall that, on a d-regular undirected unweighted graph $G = (V, E)$, the edge expansion $\Phi(S)$ of $S \subseteq V$ is defined as

$$\Phi(S) = \frac{|E(S, V \setminus S)|}{d \min\{|S|, |V \setminus S|\}}$$

where $E(S, V \setminus S)$ is the set of edges across the cut $(S, V \setminus S)$. The small set expansion problem $\mathrm{SSE}(\delta, \eta)$, where η, δ are two parameters that lie in (0, 1), can be defined as follows.

▶ **Definition 1** ($\mathsf{SSE}(\delta, \eta)$)**.** Given a regular graph $G = (V, E)$, distinguish between:
- *(Completeness)* There exists $S \subseteq V$ of size $\delta|V|$ such that $\Phi(S) \leqslant \eta$.
- *(Soundness)* For every $S \subseteq V$ of size $\delta|V|$, $\Phi(S) \geqslant 1 - \eta$.

For simplicity, we always assume that the input graphs of SSE are regular and unweighted and defer the treatment of arbitrary weighted graphs to the full version.

Roughly speaking, SSEH asserts that it is NP-hard to distinguish between a graph that has a small non-expanding subset of vertices and one in which all small subsets of vertices have almost perfect edge expansion. More formally, the hypothesis can be stated as follows.

▶ **Conjecture 1** (SSEH [32])**.** *For every $\eta > 0$, there is $\delta > 0$ such that SSE(δ, η) is NP-hard.*

Interestingly, SSEH not only implies UGC [32], but it is also equivalent to a stregthened version of the latter, in which the graph is required to have almost perfect small set expansion [33].

Since its proposal, SSEH has been used as a starting point for proving inapproximability of many problems whose hardnesses are not known otherwise. Most relevant to us is the work of Raghavendra, Steurer and Tulsiani (henceforth RST) [33] who devised a technique that exploited structures of SSE instances to avoid locality in reductions. In doing so, they obtained hardness of approximation for Min Bisection, Balanced Separator, and Minimum Linear Arrangement; these problems are not known to be hard to approximate under UGC.

1.1 Maximum Edge Biclique and Maximum Balanced Biclique

Our first result is adapting RST technique to prove inapproximability results for Maximum Edge Biclique (MEB) and Maximum Balanced Biclique (MBB). For both problems, the input is a bipartite graph. The goal for the former is to find a complete bipartite subgraph that contains as many edges as possible whereas, for the latter, the goal is to find a balanced complete bipartite subgraph that contains as many vertices as possible.

Both problems are NP-hard. MBB was stated (without proof) to be NP-hard in [12, page 196]; several proofs of this exist such as one provided in [17]. For MEB, it was proved to be NP-hard more recently by Peeters [31]. Unfortunately, much less is known when it comes to approximability of both problems. Similar to Maximum Clique, folklore algorithms give $O(n/\operatorname{polylog} n)$ approximation ratio for both MBB and MEB, and no better algorithm is known. However, not even NP-hardness of approximation of some constant ratio is known for the problems. This is in stark contrast to Maximum Clique for which strong inapproximability

results are known [14, 18, 22, 39]. Fortunately, the situation is not completely hopeless as the problems are known to be hard to approximate under stronger complexity assumptions.

Feige [10] showed that, assuming that random 3SAT formulae cannot be refuted in polynomial time, both problems[1] cannot be approximated to within n^{ε} of the optimum in polynomial time for some $\varepsilon > 0$. Later, Feige and Kogan [11] proved $2^{(\log n)^{\varepsilon}}$ ratio inapproximability for both problems for some $\varepsilon > 0$, assuming that 3SAT $\notin$ DTIME($2^{n^{3/4+\delta}}$) for some $\delta > 0$. Moreover, Khot [20] showed, assuming 3SAT $\notin$ BPTIME($2^{n^{\delta}}$) for some $\delta > 0$, that no polynomial time algorithm achieves n^{ε}-approximation for MBB for some $\varepsilon > 0$. Ambühl et al. [1] subsequently built on Khot's result and showed a similar hardness for MEB. Recently, Bhangale et al. [6] proved that both problems are hard to approximate to within[2] $n^{1-\varepsilon}$ factor for every $\varepsilon > 0$, assuming a certain strengthened version of UGC and NP $\neq$ BPP. In addition, while not stated explicitly, the author's recent reduction for Densest k-Subgraph [25] yields $n^{1/\operatorname{polyloglog} n}$ ratio inapproximability for both problems under the Exponential Time Hypothesis [16] (3SAT $\notin$ DTIME($2^{o(n)}$)) and this ratio can be improved to $n^{f(n)}$ for any $f \in o(1)$ under the stronger Gap Exponential Time Hypothesis [8, 26] (no $2^{o(n)}$ time algorithm can distinguish a fully satisfiable 3SAT formula from one which is only $(1-\varepsilon)$-satisfiable for some $\varepsilon > 0$); these ratios are better than those in [11] but worse than those in [20, 1, 6].

In this work, we prove strong inapproximability results for both problems, assuming SSEH:

▶ **Theorem 2.** *Assuming SSEH, there is no polynomial time algorithm that approximates MEB or MBB to within $n^{1-\varepsilon}$ factor of the optimum for every $\varepsilon > 0$, unless NP $\subseteq$ BPP.*

We note that the only part of the reduction that is randomized is the gap amplification via randomized graph product [5, 7]. If one is willing to assume only that NP $\neq$ P (and SSEH), our reduction still implies that both are hard to approximate to within any constant factor.

Only Bhangale et al.'s result [6] and our result achieve the inapproximability ratio of $n^{1-\varepsilon}$ for every $\varepsilon > 0$; all other results achieve at most n^{ε} ratio for some $\varepsilon > 0$. Moreover, only Bhangale et al.'s reduction and ours are candidate NP-hardness reductions, whereas each of the other reductions either uses superpolynomial time [11, 20, 1, 25] or relies on an average-case assumption [10]. It is also worth noting here that, while both Bhangale et al.'s result and our result are based on assumptions which can be viewed as stronger variants of UGC, the two assumptions are incomparable and, to the best of our knowledge, Bhangale et al.'s technique does not apply to SSEH. Due to space constraint, we defer a more in-depth discussion on the differences between the two assumptions to a longer version of this work.

Along the way, we prove inapproximability of the following hypergraph bisection problem, which may be of independent interest: given a hypergraph $H = (V_H, E_H)$ find a bisection[3] (T_0, T_1) of V_H such that the number of uncut hyperedges is maximized. We refer to this problem as *Max UnCut Hypergraph Bisection* (MUCHB). Roughly speaking, we show that, assuming SSEH, it is hard to distinguish a hypergraph whose optimal bisection cuts only ε fraction of hyperedges from one in which every bisection cuts all but ε fraction of hyperedges:

[1] While Feige only stated this for MBB, the reduction clearly works for MEB too.
[2] In [6], the inapproximability ratio is only claimed to be n^{ε} for *some* $\varepsilon > 0$. However, it is not hard to see that their result in fact implies $n^{1-\varepsilon}$ factor hardness of approximation as well.
[3] (T_0, T_1) is a bisection of V_H if $|T_0| = |T_1| = |V_H|/2, T_0 \cap T_1 = \emptyset$ and $V_H = T_0 \cup T_1$.

▶ **Lemma 3.** *Assuming SSEH, for every $\varepsilon > 0$, it is NP-hard to, given a hypergraph $H = (V_H, E_H)$, distinguish between the following two cases:*

- (Completeness) *There is a bisection (T_0, T_1) of V_H s.t.*

$$|E_H(T_0)|, |E_H(T_1)| \geqslant (1/2 - \varepsilon)|E_H|\,.$$

- (Soundness) *For every set $T \subseteq V_H$ of size at most $|V_H|/2$, $|E_H(T)| \leqslant \varepsilon |E_H|$.*

Here $E_H(T) \triangleq \{e \in E_H \mid e \subseteq T\}$ denotes the set of hyperedges inside of the set $T \subseteq V_H$.

Our result above is similar to Khot's quasi-random PCP [20]; roughly speaking, Khot's result states that it is hard (if 3SAT $\notin \bigcap_{\delta>0}$ BPTIME(2^{n^δ})) to distinguish between a d-uniform hypergraph where $1/2^{d-2}$ fraction of hyperedges are uncut in the optimal bisection from one where roughly $1/2^{d-1}$ fraction of hyperedges are uncut in any bisection. In this sense, [20] provides better soundness at the expense of worse completeness compared ours.

1.2 Minimum k-Cut

In addition to the above biclique problems, we prove an inapproximability result for the Minimum k-Cut problem, in which a weighted graph is given and the goal is to find a set of edges with minimum total weight whose removal paritions the graph into (at least) k connected components. For any fixed k, the problem was proved to be in P by Goldschmidt and Hochbaum [13], who also showed that, when k is part of the input, the problem is NP-hard. To circumvent this, Saran and Vazirani [35] devised two simple polynomial time $(2 - 2/k)$-approximation algorithms for the problem. In the ensuing years, different approximation algorithms [29, 38, 34, 37] have been proposed for the problem, none of which are able achieve an approximation ratio of $(2-\varepsilon)$ for some $\varepsilon > 0$. In fact, Saran and Vazirani themselves conjectured that $(2-\varepsilon)$-approximation is intractible for the problem [35]. In this work, we show that their conjecture is indeed true, if the SSEH holds:

▶ **Theorem 4.** *Assuming SSEH, it is NP-hard to approximate Minimum k-Cut to within $(2-\varepsilon)$ factor of the optimum for every constant $\varepsilon > 0$.*

Note that the problem was claimed to be APX-hard in [35]. However, to the best of our knowledge, the proof has never been published and no other inapproximability is known.

2 Inapproximability of Minimum k-Cut

We now proceed to prove our main results. Let us start with the simplest: Minimum k-Cut.

Proof of Theorem 4. The reduction from SSE(δ, η) to Minimum k-Cut is simple; the graph G remains the input graph for Minimum k-Cut and we let $k = \delta n + 1$ where $n = |V|$.

Completeness. If there is $S \subseteq V$ of size δn such that $\Phi(S) \leqslant \eta$, then we partition the graph into k groups where the first group is $V \setminus S$ and each of the other groups contains one vertex from S. The edges cut are the edges in $E(S, V \setminus S)$ and the edges within the set S itself. There are $d|S|\Phi(S) \leqslant \eta d|S|$ edges of the former type and only at most $d|S|/2$ of the latter. Hence, the number of edges cut in this partition is at most $(1/2 + \eta)d|S| = (1/2 + \eta)\delta dn$.

Soundness. Suppose that, for every $S \subseteq V$ of size δn, $\Phi(S) \geqslant 1-\eta$. Let $T_1, \ldots, T_k \subseteq V$ be any k-partition of the graph. Assume w.l.o.g. that $|T_1| \leqslant \cdots \leqslant |T_k|$. Let $A = T_1 \cup \cdots \cup T_i$ where i is the maximum index such that $|T_1 \cup \cdots \cup T_i| \leqslant \delta n$.

We claim that $|A| \geqslant \delta n - \sqrt{n}$. To see that this is the case, suppose for the sake of contradiction that $|A| < \delta n - \sqrt{n}$. Since $|A \cup T_{i+1}| > \delta n$, we have $T_{i+1} > \sqrt{n}$. Moreover, since $A = T_1 \cup \cdots T_i$, we have $i \leqslant |A| < \delta n - \sqrt{n}$. As a result, we have $n = |T_1 \cup \cdots \cup T_k| \geqslant |T_{i+1} \cup \cdots \cup T_k| \geqslant (k-i)|T_i| > \sqrt{n} \cdot \sqrt{n} = n$, which is a contradiction. Hence, $|A| \geqslant \delta n - \sqrt{n}$.

Now, note that, for every $S \subseteq V$ of size δn, $\Phi(S) \geqslant 1-\eta$ implies that $|E(S)| \leqslant \eta d \delta n/2$ where $E(S)$ denote the set of all edges within S. Since $|A| \leqslant \delta n$, we also have $|E(A)| \leqslant \eta d \delta n/2$. As a result, the number of edges in the cut $(A, V \setminus A)$ is at least

$$d|A| - \eta d \delta n \geqslant (1-\eta) d \delta n - d\sqrt{n} = \left(1 - \eta - \frac{1}{\delta\sqrt{n}}\right) \delta d n.$$

For every constant $\varepsilon > 0$, by setting $\eta = \varepsilon/20$ and $n > 100/(\varepsilon^2\delta^2)$, the ratio between the two cases is at least $(2-\varepsilon)$, which concludes the proof of Theorem 4. ◀

3 Inapproximability of MEB and MBB

Let us now turn our attention to MEB and MBB. First, note that we can reduce MUCHB to MEB/MBB by just letting the two sides of the bipartite graph be E_H and creating an edge (e_1, e_2) iff $e_1 \cap e_2 = \emptyset$. This immediately shows that Lemma 3 implies the following:

▶ **Lemma 5.** *Assuming SSEH, for every $\delta > 0$, it is NP-hard to, given a bipartite graph $G = (L, R, E)$ with $|L| = |R| = n$, distinguish between the following two cases:*

- (Completeness) *G contains $K_{(1/2-\delta)n,(1/2-\delta)n}$ as a subgraph.*
- (Soundness) *G does not contain $K_{\delta n,\delta n}$ as a subgraph.*

Here $K_{t,t}$ denotes the complete bipartite graph in which each side contains t vertices.

Note that Theorem 2 follows from Lemma 5 by gap amplification via randomized graph product [5, 7]. Since this has been analyzed before even for biclique [20, Appendix D], we do not repeat the argument here.

We are now only left to prove Lemma 3; we devote the rest of this section to this task.

3.1 Preliminaries

Before we continue, we need additional notations and preliminaries. For every graph G and every vertex v, we will write $G(v)$ to denote the uniform distribution on its neighbors.

It will be convenient for us to use a different (but equivalent) formulation of SSEH. To state it, we will define a variant of SSE(δ, η) called SSE(δ, η, M); the completeness remains the same whereas the soundness is strengthened to include all S of size between $\frac{\delta|V|}{M}$ and $\delta|V|M$.

▶ **Definition 6** (SSE(δ, η, M))**.** Given a regular graph $G = (V, E)$, distinguish between:

- *(Completeness)* There exists $S \subseteq V$ of size $\delta|V|$ such that $\Phi(S) \leqslant \eta$.
- *(Soundness)* For every $S \subseteq V$ with $|S| \in \left[\frac{\delta|V|}{M}, \delta|V|M\right]$, $\Phi(S) \geqslant 1-\eta$.

The new formulation of the hypothesis can now be stated as follows.

▶ **Conjecture 2.** *For every $\eta, M > 0$, there is $\delta > 0$ such that SSE(δ, η, M) is NP-hard.*

Raghavendra et al. [33] showed that this formulation is equivalent to the earlier formulation (Conjecture 1); please refer to Appendix A.2 of [33] for more details of the proof.

While our reduction can be understood without notation of unique games, it is best described in a context of unique games reductions. We provide a definition of unique games below.

▶ **Definition 7** (Unique Game (UG)). A unique game instance $(\mathcal{G}, [R], \{\pi_e\}_{e\in E})$ consists of a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a label set $[R] = \{1, \dots, R\}$, and, for each $e \in \mathcal{E}$, a permutation $\pi_e : [R] \to [R]$. The goal is to find an assignment $F : V \to [R]$ such that, for as many edges $(u, v) \in \mathcal{E}$ as possible, we have $\pi_{(u,v)}(F(u)) = F(v)$; these edges are said to be *satisfied*.

Khot's UGC [19] states that, for every $\varepsilon > 0$, it is NP-hard to distinguish between a unique game in which there exists an assignment satisfying at least $(1 - \varepsilon)$ fraction of edges from one in which every assignment satisfies at most ε fraction of edges.

Finally, we need some preliminaries in discrete Fourier analysis. We state here only few facts that we need. We refer interested readers to [30] for more details about the topic.

For any discrete probability space Ω, $f : \Omega^R \to [0, 1]$ can be written as $\sum_{\sigma\in[|\Omega|]^R} \hat{f}(\sigma)\phi_\sigma$ where $\{\phi_\sigma\}_{\sigma\in[|\Omega|]^R}$ is the product Fourier basis of $L^2(\Omega^R)$ (see [30, Chapter 8.1]). The degree-d influence on the j-th coordinate of f is $\mathrm{infl}_j^d(f) \triangleq \sum_{\sigma\in[|\Omega|]^R, \sigma_j\neq 1, \#\sigma\leqslant d} \hat{f}^2(\sigma)$ where $\#\sigma \triangleq |\{i \in [R] \mid \sigma_i \neq 1\}|$. It is well known that $\sum_{j=1}^R \mathrm{infl}_j^d(f) \leqslant d$ (see [28, Proposition 3.8]).

We also need the following theorem. It follows easily[4] from the so-called "It Ain't Over Till It's Over" conjecture, which is by now a theorem [28, Theorem 4.9].

▶ **Theorem 8** ([28]). *For any $\beta, \varepsilon_T, \gamma > 0$, there exists $\kappa > 0$ and $t, d \in \mathbb{N}$ such that, if any functions $f_1, \dots, f_t : \Omega^R \to \{0, 1\}$ where Ω is a probability space whose probability of each atom is at least β satisfy*[5]

$$\forall i \in [t], \mathop{\mathbb{E}}_{x\in\Omega^R}[f_i(x)] \leqslant 0.99 \quad \text{and} \quad \forall j \in [R], \forall 1 \leqslant i_1 \neq i_2 \leqslant t, \min\{\mathrm{infl}_j^d(f_{i_1}), \mathrm{infl}_j^d(f_{i_2})\} \leqslant \kappa,$$

then

$$\Pr_{x\in\Omega^R, D\sim S_{\varepsilon_T}(R)}\left[\bigwedge_{i=1}^t f_i(C_D(x)) \equiv 1\right] < \gamma$$

where $D \sim S_{\varepsilon_T}(R)$ is a random subset of $[R]$ where each $i \in [R]$ is included independently w.p. ε_T, $C_D(x) \triangleq \{x' \mid x'_{[R]\setminus D} = x_{[R]\setminus D}\}$ and $f_i(C_D(x)) \equiv 1$ is short for $\forall x' \in C_D(x), f_i(x') = 1$.

3.2 Bansal-Khot Long Code Test and A Candidate Reduction

Theorem 8 leads us nicely to the Bansal-Khot long code test [4]. For UGC hardness reductions, one typically needs a *long code test* (aka dictatorship gadget) which, on input $f_1, \dots, f_t : \{0, 1\}^R \to \{0, 1\}$, has the following properties:

- *(Completeness)* If $f_1 = \cdots = f_t$ is a long code[6], the test accepts with large probability.
- *(Soundness)* If $f_1, \dots, f_t$ are balanced (i.e. $\mathbb{E} f_1 = \cdots = \mathbb{E} f_t = 1/2$) and are "far from being a long code", then the test accepts with low probability.

  A widely-used notion of "far from being a long code", and one we will use here, is that the functions do not share a coordinate with large low degree influence (i.e. for every $j \in [R]$ and every $i_1 \neq i_2 \in [t]$, at least one of $\mathrm{infl}_j^d(f_{i_1})$, $\mathrm{infl}_j^d(f_{i_2})$ is small),

[4] For more details on how this version follows from there, please refer to [36, page 769].
[5] 0.99 could be replaced by any constant less than one; we use it to avoid introducing more parameters.
[6] A long code is simply j-junta (i.e. a function that depends only on the x_j) for some $j \in [R]$.

Input: A unique game $(\mathcal{G} = (\mathcal{V}, \mathcal{E}), [R], \{\pi_e\}_{e\in E})$ and parameters $\ell \in \mathbb{N}$ and $\varepsilon_T \in (0, 1)$.
Output: A hypergraph $H = (V_H, E_H)$.
The vertex set V_H is $\mathcal{V} \times \{0, 1\}^R$ and the hyperedges are distributed as follows:

- Sample $u \sim \mathcal{V}$ and sample $v_1 \sim \mathcal{G}(u), \ldots, v_\ell \sim \mathcal{G}(u)$.
- Sample $x \sim \{0, 1\}^R$ and a subset $D \sim S_{\varepsilon_T}(R)$.
- Output a hyperedge $e = \{(v_p, x') \mid p \in [\ell], x' \in C_D(x)\}$.

Figure 1 A Candidate Reduction from UG to MUCHB.

Bansal and Khot's long code test works as follows: pick $x \sim \{0, 1\}^R$ and $D \sim S_{\varepsilon_T}(R)$. Then, test whether f_i evaluates to 1 on the whole $C_D(x)$. Note that this can be viewed as an "algorithmic" version of Theorem 8; more specifically, the theorem (with $\Omega = \{0, 1\}$) immediately implies the soundness property of this test. On the other hand, it is obvious that, if $f_1 = \cdots = f_t$ is a long code, then the test accepts with probability $1/2 - \varepsilon_T$.

Bansal and Khot used this test to prove tight hardness of approximation of Vertex Cover. The reduction is via a natural composition of the test with unique games. Their reduction also gives a cadidate reduction from UG to MUCHB, which is stated below in Figure 1.

As is typical for gadget reductions, for $T \subseteq V_H$, we view the indicator function $f_u(x) \triangleq \mathbb{1}[(u, x) \in T]$ for each $u \in \mathcal{V}$ as the intended long code. If there exists an assignment ϕ to the unique game instance that satisfies nearly all the constraints, then the bisection corresponding to $f_u(x) = x_{\phi(u)}$ cuts only small fraction of edges, which yields the completeness of MUCHB.

As for the soundness, we would like to decode an UG assignment from $T \subseteq V_H$ of size at most $|V_H|/2$ which contains at least ε fraction of hyperedges. In terms of the tests, this corresponds to a collection of functions $\{f_u\}_{u\in\mathcal{V}}$ such that $\mathbb{E}_{u\sim\mathcal{V}} \mathbb{E}_{x\sim\{0,1\}^R} f_u(x) = 1/2$ and the Bansal-Khot test on $f_{v_1}, \ldots, f_{v_t}$ passes with probability at least ε where $v_1, \ldots, v_t$ are sampled as in Figure 1. Now, if we assume that $\mathbb{E}_x f_u(x) \leqslant 0.99$ for all $u \in \mathcal{V}$, then such decoding is possible via a similar method as in [4] since Theorem 8 can be applied here.

Unfortunately, the assumption $\mathbb{E}_x f_u(x) \leqslant 0.99$ does not hold for an arbitrary $T \subseteq V_H$ and the soundness property indeed fails. For instance, imagine the constraint graph $\mathcal{G}$ of the starting unique game instance consisting of two disconnected components of equal size; let $\mathcal{V}_0$ and $\mathcal{V}_1$ be the set of vertices in the two components. In this case, if we set $T_0 = \mathcal{V}_0$ and $T_1 = \mathcal{V}_1$, then the bisection (T_0, T_1) does not even cut a single edge! This is regardless of whether there exists an assignment to the UG that satisfies a large fraction of edges.

3.3 RST Technique and The Reduction from SSE to MUCHB

The issue described above is common for graph problems that involves some form of expansion of the graph. The RST technique [33] was indeed invented to specifically circumvent this issue. It works by first reducing SSE to UG and then exploiting the structure of the constructed UG instance when composing it with a long code test; this allows them to avoid extreme cases such as one above. There are four parameters in the reduction: $R, k \in \mathbb{N}$ and ε_V, β. Before we describe the reduction, let us define additional notations here:

- Let $G^{\otimes R}$ denote the R-tensor graph of G; the vertex set of $G^{\otimes R}$ is V^R and there is an edge between A, B if and only if there is an edge between A_i, B_i in G for every $i \in [R]$.
- For each $A \in V^R$, $T_V(A)$ denote the distribution on V^R where the i-th coordinate is set to A_i with probability $1 - \varepsilon_V$ and is randomly sampled from V otherwise.

- Let $\Pi_{R,k}$ denote the set of all permutations π's of $[R]$ such that, for each $j \in [k]$, $\pi(\{R(j-1)/k+1, \ldots, Rj/k\}) = \{R(j-1)/k+1, \ldots, Rj/k\}$.
- Let $\{0,1,\perp\}_\beta$ denote the probability space such that the probability for $0,1$ are both $\beta/2$ and the probability for $\perp$ is $1-\beta$.

The first step of reduction takes an $\mathrm{SSE}(\delta, \eta, M)$ instance $G = (V, E)$ and produces a unique game $\mathcal{U} = (\mathcal{G} = (\mathcal{V}, \mathcal{E}), [R], \{\pi_e\}_{e \in E})$ where $\mathcal{V} = V^R$ and the edges are distributed as follows:

1. Sample $A \sim V^R$ and $\tilde{A} \sim T_V(A)$.
2. Sample $B \sim G^{\otimes R}(\tilde{A})$ and $\tilde{B} \sim T_V(B)$.
3. Sample two random $\pi_A, \pi_B \sim \Pi_{R,k}$.
4. Output an edge $e = (\pi_A(\tilde{A}), \pi_B(\tilde{B}))$ with $\pi_{(A,B)} = \pi_B \circ \pi_A^{-1}$.

Here ε_V is a small constant, k is large and R/k should be think of as $\Theta(1/\delta)$. When there exists a set $S \subseteq V$ of size $\delta|V|$ with small edge expansion, the intended assignment is to, for each $A \in V^R$, find the first block $j \in [k]$ such that $|A(j) \cap S| = 1$ where $A(j)$ denotes the multiset $\{A_{R(j-1)/k+1}, \ldots, A_{Rj/k}\}$ and let $F(A)$ be the coordinate of the vertex in that intersection. If no such j exists, we assign $F(A)$ arbitrarily. Note that, since $R/k = \Theta(1/\delta)$, $\Pr[|A(j) \cap S| = 1]$ is constant, which means that only $2^{-\Omega(k)}$ fraction of vertices are assigned arbitrarily. Moreover, it is not hard to see that, for the other vertices, their assignments rarely violate constraints as ε_V and $\Phi(S)$ are small. This yields the completeness. In addition, the soundness was shown in [32, 33], i.e., if every $S \subseteq V$ of size $\delta|V|$ has near perfect expansion, no assignment satisfies many constraints in $\mathcal{U}$ (see Lemma 13).

The second step is to reduce this UG instance to a hypergraph $H = (V_H, E_H)$. Instead of making the vertex set $V^R \times \{0,1\}^R$ as in the previous candidate reduction, we will instead make $V_H = V^R \times \Omega^R$ where $\Omega = \{0,1,\perp\}_\beta$ and β is a small constant. This does not seem to make much sense from the UG reduction standpoint because we typically want to assign which side of the bisection $(A, x) \in V_H$ is in according to $x_{F(A)}$ but $x_{F(A)}$ could be $\perp$ in this construction. However, it makes sense when we view this as a reduction from SSE directly: let us discard all coordinates i's such that $x_i = \perp$ and define $A(j, x) \triangleq \{A_i \mid i \in \{R(j-1)/k+1, \ldots, Rj/k\} \wedge x_i \neq \perp\}$. Then, let $j^*(A, x) \triangleq \min\{j \mid |A(j, x) \cap S| = 1\}$ and let $i^*(A, x)$ be the coordinate in the intersection between $A(j^*(A, x), x)$ and S, and assign (A, x) to $T_{x_{i^*(A,x)}}$. (If $j^*(A, x)$ does not exists, then assign (A, x) arbitrarily.)

Observe that, in the intended solution, the side that (A, x) is assigned to does not change if (1) A_i is modified for some $i \in [R]$ s.t. $x_i = \perp$ or (2) we apply some permutation $\pi \in \Pi_{R,k}$ to both A and x. In other words, we can "merge" two vertices (A, x) and (A', x') that are equivalent through these changes together in the reduction. For notational convenience, instead of merging vertices, we will just modify the reduction so that, if (A, x) is included in some hyperedge, then every (A', x') reachable from (A, x) by these operations is also included in the hyperedge. More specifically, if we define $M_x(A) \triangleq \{A' \in V^R \mid A'_i = A_i \text{ for all } i \in [R] \text{ such that } x_i \neq \perp\}$ corresponding to the first operation, then we add $\pi(A', x)$ to the hyperedge for every $A' \in M_x(A)$ and $\pi \in \Pi_{R,k}$. The full reduction is shown in Figure 2.

Note that the test we apply here is slightly different from Bansal-Khot test as our test is on $\Omega = \{0,1,\perp\}_\beta$ instead of $\{0,1\}$ used in [4]. Another thing to note is that now our vertices and hyperedges are weighted, the vertices according to the product measure of $V^R \times \Omega^R$ and the edges according to the distribution produced from the reduction. We write μ_H to denote the measure on the vertices, i.e., for $T \subseteq V^R \times \{0,1,\perp\}^R$, $\mu_H(T) = \Pr_{A \sim V^R, x \sim \Omega^R}[(A, x) \in T]$, and we abuse the notation $E_H(T)$ and use it to denote the probability that a hyperedge as generated in Figure 2 lies completely in T. We note here that, while the MUCHB as stated

Input: A graph G with vertex set V and parameters $R, k, \ell \in \mathbb{N}$ and $\varepsilon_T, \varepsilon_V, \beta \in (0,1)$.
Output: A hypergraph $H = (V_H, E_H)$.
$V_H \triangleq V^R \times \Omega^R$ where $\Omega \triangleq \{0, 1, \perp\}_\beta$ and the hyperedges are distributed as follows:

- Sample $A \sim V^R$ and $\tilde{A}^1, \dots, \tilde{A}^\ell \sim T_V(A)$.
- Sample $B^1 \sim G^{\otimes R}(\tilde{A}^1), \dots, B^\ell \sim G^{\otimes R}(\tilde{A}^\ell)$ and $\tilde{B}^1 \sim T_V(B^1), \dots, \tilde{B}^\ell \sim T_V(B^\ell)$
- Sample $x \in \Omega^R$ and a subset $D \sim S_{\varepsilon_T}(R)$.
- Output a hyperedge $e = \{\pi(B', x') \mid p \in [\ell], \pi \in \Pi_{R,k}, x' \in C_D(x), B' \in M_{x'}(\tilde{B}^p)\}$.

Figure 2 Reduction from SSE to Max UnCut Hypergraph Bisection.

in Lemma 3 is unweighted, it is not hard to see that we can go from weighted version to unweighted by copying each vertex and each edge proportional[7] to their weights.

The advantage of this reduction is that the vertex "merging" makes gadget reduction non-local; for instance, it is clear that even if the starting graph V has two connected components, the resulting hypergraph is now connected. In fact, Raghavendra et al. [33] show a much stronger quantitative bound. To state this, let us consider any $T \in V_H$ with $\mu_H(T) = 1/2$. From how the hyperedges are defined, we can assume w.l.o.g. that, if $(A, x) \in T$, then $\pi(A', x) \in T$ for every $A' \in M_x(A)$ and every $\pi \in \Pi_{R,k}$. Again, let $f_A(x) \triangleq \mathbb{1}[(A, x) \in T]$. The following bound on the variance of $\mathbb{E}_x f_A(x)$ is implied by the proof of Lemma 6.6 in [33]:

$$\mathop{\mathbb{E}}_{A \sim V^R} \left(\mathop{\mathbb{E}}_{x \sim \Omega^R} f_A(x) - 1/2 \right)^2 \leqslant \beta.$$

The above bound implies that, for most A's, the mean of f_A cannot be too large. This will indeed allow us to ultimately apply Theorem 8 on a certain fraction of the tuples $(\tilde{B}^1, \dots, \tilde{B}^\ell)$ in the reduction, which leads to an UG assignment with non-negligible value.

3.4 Completeness

In the completeness case, we define a bisection similar to that described above. This bisection indeed cuts only a small fraction of hyperedges; quantitatively, this yields the following lemma. Since its proof consists mainly of calculations, we omit it from this extended abstract.

▶ **Lemma 9.** *If there is a set $S \subseteq V$ such that $\Phi(S) \leqslant \eta$ and $|S| = \delta|V|$ where $\delta \in \left[\frac{k}{10\beta R}, \frac{k}{\beta R}\right]$, then there is a bisection (T_0, T_1) of V_H such that $E_H(T_0), E_H(T_1) \geqslant 1/2 - O(\varepsilon_T/\beta) - O(\eta\ell/\beta) - O(\varepsilon_V \ell/\beta) - 2^{-\Omega(k)}$ where $O(\cdot)$ and $\Omega(\cdot)$ hide only absolute constants.*

3.5 Soundness

Let us consider any set T such that $\mu_H(T) \leqslant 1/2$. We would like to give an upper bound on $E_H(T)$. From how we define hyperedges, we can assume w.l.o.g. that $(A, x) \in T$ if and only if $\pi(A', x) \in T$ for every $A' \in M_x(A)$ and $\pi \in \Pi_{R,k}$. We call such T $\Pi_{R,k}$*-invariant*.

Let $f : V^R \times \Omega^R \to \{0, 1\}$ denote the indicator function for T, i.e., $f(A, x) = 1$ if and only if $(A, x) \in T$. Note that $\mathbb{E}_{A \sim V^R, x \sim \Omega^R} f(A, x) = \mu_H(T) \leqslant 1/2$. Following notation from [33], we write $f_A(x)$ as a shorthand for $f(A, x)$. In addition, for each $A \in V^R$, we will write $\tilde{B} \sim \Gamma(A)$ as a shorthand for $\tilde{B}$ generated randomly by sampling $\tilde{A} \sim T_V(A)$, $B \sim G^{\otimes R}(\tilde{A})$

[7] Note that this is doable since we can pick $\beta, \varepsilon_V, \varepsilon_T$ to be rational.

and $\tilde{B} \sim T_V(B)$ respectively. Let us restate Raghavendra et al.'s [33] Lemma regarding the variance of $\mathbb{E}_x f_A(x)$ in a more convenient formulation below.

▶ **Lemma 10** ([33, Lemma 6.6][8]). *For every $A \in V^R$, let $\mu_A \triangleq \mathbb{E}_{x\sim\Omega^R} f_A(x)$. We have*

$$\mathop{\mathbb{E}}_{A\sim V^R}\left(\mathop{\mathbb{E}}_{\tilde{B}\sim\Gamma(A)}\mu_{\tilde{B}} - \mu_H(T)\right)^2 \leqslant \beta.$$

To see how the above lemma helps us decode an UG assignment, observe that, if our test accepts on $f_{\tilde{B}^1}, \ldots, f_{\tilde{B}^\ell}, x, D$, then it also accepts on any subset of the functions (with the same x, D); hence, to apply Theorem 8, it suffices that t of the functions have means $\leqslant 0.99$. We will choose ℓ to be large compared to t. Using above lemma and a standard tail bound, we can argue that Theorem 8 is applicable for almost all tuples $\tilde{B}^1, \ldots, \tilde{B}^\ell$, as stated below. Due to space constraint, we omit its proof from this extended abstract.

▶ **Lemma 11.** *For any positive integer $t \leqslant 0.01\ell$,*

$$\Pr_{A\sim V^R, \tilde{B}^1,\ldots,\tilde{B}^\ell\sim\Gamma(A)}[|\{i \in [\ell] \mid \mu_{\tilde{B}^i} \leqslant 0.99\}| \geqslant t] \geqslant 1 - 10\beta - 2^{-\ell/100}.$$

3.5.1 Decoding an Unique Games Assignment

With Lemma 11 ready, we can now decode an UG assignment via a similar technique from [4].

▶ **Lemma 12.** *For any $\varepsilon_T, \gamma, \beta > 0$, let $t = t(\varepsilon_T, \gamma, \beta), \kappa = \kappa(\varepsilon_T, \gamma, \beta)$ and $d = d(\varepsilon_T, \gamma, \beta)$ be as in Theorem 8. For any integer $\ell \geqslant 100t$, if there exists $T \subseteq V_H$ of such that $\mu_H(T) \leqslant 1/2$ and $E_H(T) \geqslant 2\gamma + 10\beta + 2^{-\ell/100}$, then there exists $F : V^R \to [R]$ such that*

$$\Pr_{A\sim V^R, \tilde{B}\sim\Gamma(A), \pi_A, \pi_B\sim\Pi_{R,k}}[\pi_A^{-1}(F(\pi_A(\tilde{A}))) = \pi_B^{-1}(F(\pi_B(\tilde{B})))] \geqslant \frac{\gamma\kappa^2}{4d^2\ell^2}.$$

Proof. The decoding procedure is as follows. For each $A \in V^R$, we construct a set of candidate labels $\mathrm{Cand}[A] \triangleq \{j \in [R] \mid \mathrm{infl}_j^d(f_A) \geqslant \kappa\}$. We generate F randomly by, with probability $1/2$, setting $F(A)$ to be a random element of $\mathrm{Cand}[A]$ and, with probability $1/2$, sampling $\tilde{B} \sim \Gamma(A)$ and setting $F(A)$ to be a random element from $\mathrm{Cand}[B]$. Note that, if the candidate set is empty, then we simply pick an arbitrary assignment.

From our assumption that T is $\Pi_{R,k}$-invariant, it follows that, for every $A \in V^R, \pi \in \Pi_{R,k}$ and $j \in [R]$, $\Pr_F[\pi^{-1}(F(\pi(A))) = j] = \Pr_F[F(A) = j]$. In other words, we have

$$\begin{aligned}\Pr_{F,A\sim V^R, \tilde{B}\sim\Gamma(A), \pi_A, \pi_B\sim\Pi_{R,k}}&[\pi_A^{-1}(F(\pi_A(\tilde{A}))) = \pi_B^{-1}(F(\pi_B(\tilde{B})))] \\ &= \Pr_{F,A\sim V^R, \tilde{B}\sim\Gamma(A)}[F(\tilde{A}) = F(\tilde{B})]. \qquad (1)\end{aligned}$$

Next, note that, from how our reduction is defined, $E_H(T)$ can be written as

$$E_H(T) = \Pr_{A\sim V^R, \tilde{B}^1,\ldots,\tilde{B}^\ell\sim\Gamma(A), x\sim\Omega^R, D\sim S_{\varepsilon_T}(R)}\left[\bigwedge_{i=1}^{\ell} f_{\tilde{B}^i}(C_D(x)) \equiv 1\right].$$

[8] Lemma 6.6 in [33] involves symmetrizing f's, but we do not need it here since T is $\Pi_{R,k}$-invariant.

From $E_H(T) \geqslant 2\gamma + 10\beta + 2^{-\ell/100}$ and from Lemma 11, we can conclude that

$$\Pr_{A,\tilde{B}^1,\ldots,\tilde{B}^\ell,x,D}\left[\left(\bigwedge_{i=1}^{\ell} f_{\tilde{B}^i}(C_D(x)) \equiv 1\right) \wedge \left(|\{i \in [\ell] \mid \mu_{\tilde{B}^i} \leqslant 0.99\}| \geqslant t\right)\right] \geqslant 2\gamma.$$

From Markov's inequality, we have

$$\begin{aligned}\gamma &\leqslant \Pr_{A,\tilde{B}^1,\ldots,\tilde{B}^\ell}\left[\Pr_{x,D}\left[\left(\bigwedge_{i=1}^{\ell} f_{\tilde{B}^i}(C_D(x)) \equiv 1\right) \wedge \left(|\{i \in [\ell] \mid \mu_{\tilde{B}^i} \leqslant 0.99\}| \geqslant t\right)\right] \geqslant \gamma\right] \\ &= \Pr_{A,\tilde{B}^1,\ldots,\tilde{B}^\ell}\left[\left(\Pr_{x,D}\left[\bigwedge_{i=1}^{\ell} f_{\tilde{B}^i}(C_D(x)) \equiv 1\right] \geqslant \gamma\right) \wedge \left(|\{i \in [\ell] \mid \mu_{\tilde{B}^i} \leqslant 0.99\}| \geqslant t\right)\right].\end{aligned}$$

A tuple $(A, \tilde{B}^1, \ldots, \tilde{B}^\ell)$ is said to be *good* if $\Pr_{x\sim\Omega^R, D\sim S_{\varepsilon_T}(R)}\left[\bigwedge_{i=1}^{\ell} f_{\tilde{B}^i}(C_D(x)) \equiv 1\right] \geqslant \gamma$ and $|\{i \in [\ell] \mid \mu_{\tilde{B}^i} \leqslant 0.99\}| \geqslant t$. For such tuple, Theorem 8 implies that there exist $i_1 \neq i_2 \in [\ell], j \in [R]$ s.t. $\mathrm{infl}^d_j(f_{\tilde{B}^{i_1}}), \mathrm{infl}^d_j(f_{\tilde{B}^{i_2}}) \geqslant \kappa$. This means that $\mathrm{Cand}(\tilde{B}^{i_1}) \cap \mathrm{Cand}(\tilde{B}^{i_2}) \neq \emptyset$.

Hence, if we sample a tuple $(A, \tilde{B}^1, \ldots, \tilde{B}^\ell)$ at random, and then sample two different $\tilde{B}, \tilde{B}'$ randomly from $\tilde{B}^1, \ldots, \tilde{B}^\ell$, then the tuple is good with probability at least γ and, with probability $1/\ell^2$, we have $\tilde{B} = \tilde{B}^{i_1}, \tilde{B}' = \tilde{B}^{i_2}$. This gives the following bound:

$$\Pr_{A,\tilde{B},\tilde{B}'}\left[\mathrm{Cand}(\tilde{B}) \cap \mathrm{Cand}(\tilde{B}') \neq \emptyset\right] \geqslant \frac{\gamma}{\ell^2}.$$

Now, observe that $\tilde{B}$ and $\tilde{B}'$ above are distributed in the same way as if we pick both of them independently with respect to $\Gamma(A)$. Recall that, with probability $1/2$, $F(A)$ is a random element of $\mathrm{Cand}(\tilde{B})$ where $\tilde{B} \sim \Gamma(A)$ and, with probability $1/2$, $F(\tilde{B}')$ is a random element of $\mathrm{Cand}(\tilde{B}')$. Moreover, since the sum of degree d-influence is at most d [28, Proposition 3.8], the candidate sets are of sizes at most d/κ. As a result, the above bound yields

$$\Pr_{A\sim V^R, \tilde{B}'\sim\Gamma(A)}[F(A) = F(\tilde{B}')] \geqslant \frac{\gamma\kappa^2}{4d^2\ell^2},$$

which, together with (1), concludes the proof of the lemma. ◀

3.5.2 Decoding a Small Non-Expanding Set

To relate our decoded UG assignment back to a small non-expanding set in G, we use the following lemma of [33], which roughly states that, with the right parameters, the soundness case of SSEH implies that only small fraction of constriants in the UG can be satisfied.

▶ **Lemma 13** ([33, Lemma 6.11]). *If there exists $F : V^R \to [R]$ such that*

$$\Pr_{A\sim V^R, \tilde{B}\sim\Gamma(A), \pi_A, \pi_B\sim\Pi_{R,k}}[\pi_A^{-1}(F(\pi_A(\tilde{A}))) = \pi_B^{-1}(F(\pi_B(\tilde{B})))] \geqslant \zeta,$$

then there exists a set $S \subseteq V$ with $\frac{|S|}{|V|} \in \left[\frac{\zeta}{16R}, \frac{3k}{\varepsilon_V R}\right]$ with $\Phi(S) \leqslant 1 - \frac{\zeta}{16k}$.

By combining the above lemma with Lemma 12, we immediately arrive at the following:

▶ **Lemma 14.** *For any $\varepsilon_T, \gamma, \beta > 0$, let $t = t(\varepsilon_T, \gamma, \beta), \kappa = \kappa(\varepsilon_T, \gamma, \beta)$ and $d = d(\varepsilon_T, \gamma, \beta)$ be as in Theorem 8. For any integer $\ell \geqslant 100t$ and any $\varepsilon_V > 0$, if there exists $T \subseteq V_H$ with $\mu_H(T) \leqslant 1/2$ such that $E_H(T) \geqslant 2\gamma + 10\beta + 2^{-\ell/100}$, then there exists a set $S \subseteq V$ with $\frac{|S|}{|V|} \in \left[\frac{\zeta}{16R}, \frac{3k}{\varepsilon_V R}\right]$ with $\Phi(S) \leqslant 1 - \frac{\zeta}{16k}$ where $\zeta = \frac{\gamma\kappa^2}{4d^2\ell^2}$.*

3.6 Putting Things Together

We can now deduce inapproximability of MUCHB by simply picking appropriate parameters.

Proof of Lemma 3. The parameters are chosen as follows:

- Let $\beta = \varepsilon/30, \gamma = \varepsilon/6$, and $k = \Omega(\log(1/\varepsilon))$ so that the term $2^{-\Omega(k)}$ in Lemma 9 is $\leqslant \varepsilon/4$.
- Let $\varepsilon_T = O(\beta\varepsilon)$ so that the error term $O(\varepsilon_T/\beta)$ in Lemma 9 is at most $\varepsilon/4$.
- Let $t = t(\varepsilon_T, \gamma, \beta), \kappa = \kappa(\varepsilon_T, \gamma, \beta)$ and $d = d(\varepsilon_T, \gamma, \beta)$ be as in Theorem 8.
- Let $\zeta = \frac{\gamma\kappa^2}{4d^2\ell^2}$ be as in Lemma 14 and let $\ell = \max\{100t, 1000\log(1/\varepsilon)\}$.
- Let $\varepsilon_V = O(\varepsilon\beta/\ell)$ where so that the error term $O(\varepsilon_V\ell/\beta)$ in Lemma 9 is at most $\varepsilon/4$.
- Let $\eta = \min\{\frac{\zeta}{32k}, O(\varepsilon\beta/\ell)\}$ so that the error term $O(\eta\ell/\beta)$ in Lemma 9 is at most $\varepsilon/4$.
- Let $M = \max\{\frac{16k}{\beta\zeta}, \frac{3\beta}{\varepsilon_V}\}$.
- Finally, let $R = \frac{k}{\beta\delta}$ where $\delta = \delta(\eta, M)$ is the parameter from the SSEH (Conjecture 2).

Let $G = (V, E)$ be an instance of $\mathrm{SSE}(\eta, \delta, M)$ and let $H = (V_H, G_H)$ be the hypergraph resulted from our reduction. If there exists $S \subseteq V$ of size $\delta|V|$ of expansion at most η, Lemma 9 implies that there is a bisection (T_0, T_1) of V_H such that $E_H(T_0), E_H(T_1) \geqslant 1/2 - \varepsilon$.

As for the soundness, Lemma 14 with our choice of parameters implies that, if there exists a set $T \subseteq V_H$ with $\mu(T) \leqslant 1/2$ and $E_H(T_0) \geqslant \varepsilon$, there exists $S \subseteq V$ with $|S| \in \left[\frac{\delta|V|}{M}, \delta|V|M\right]$ whose expansion is less than $1-\eta$. The contrapositive of this yields the soundness property. ◀

4 Conclusion

In this work, we prove essentially tight inapproximability of MEB, MBB and Minimum k-Cut based on SSEH. Our results, expecially for the biclique problems, demonstrate further the applications of the hypothesis and particularly the RST technique [33] in proving hardness of graph problems that involve some form of expansion. An obvious but intriguing research direction is to try to utilize the technique to other problems. One plausible candidate problem to this end is the 2-Catalog Segmentation Problem [24] since a natural candidate reduction for this problem fails due to a similar counterexample as in Section 3.2.

Acknowledgement. I am grateful to Prasad Raghavendra for providing his insights on the Small Set Expansion problem and techniques developed in [32, 33] and Luca Trevisan for lending his expertise in PCPs with small free bits. I would also like to thank Haris Angelidakis for useful discussions on Minimum k-Cut and Daniel Reichman for inspiring me to work on Maximum Edge Biclique and Maximum Balanced Biclique. Finally, I thank anonymous reviewers for their useful comments and, more specifically, for pointing me to [6].

References

1 Christoph Ambühl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM J. Comput.*, 40(2):567–596, April 2011.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, January 1998.

4 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *IEEE FOCS*, pages 453–462, 2009.

5 Piotr Berman and Georg Schnitger. On the complexity of approximating the independent set problem. *Inf. Comput.*, 96(1):77–94, 1992.

6 Amey Bhangale, Rajiv Gandhi, Mohammad Taghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Bicovering: Covering edges with two small subsets of vertices. In *ICALP*, pages 6:1–6:12, 2016.

7 Avrim Blum. Algorithms for approximate graph coloring. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1991.

8 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *ECCC*, 23:128, 2016.

9 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *ACM STOC*, pages 624–633, 2014.

10 Uriel Feige. Relations between average case complexity and approximation complexity. In *ACM STOC*, pages 534–543, 2002.

11 Uriel Feige and Shimon Kogan. Hardness of approximation of the balanced complete bipartite subgraph problem. Technical report, Weizmann Institute of Science, Israel, 2004.

12 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

13 Olivier Goldschmidt and Dorit S Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of operations research*, 19(1):24–37, 1994.

14 Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *IEEE FOCS*, pages 627–636, 1996.

15 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

16 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, December 2001.

17 David S. Johnson. The NP-completeness column: An ongoing guide. *J. Algorithms*, 8(5):438–448, September 1987.

18 Subhash Khot. Improved inaproximability results for MaxClique, chromatic number and approximate graph coloring. In *IEEE FOCS*, pages 600–609, 2001.

19 Subhash Khot. On the power of unique 2-prover 1-round games. In *ACM STOC*, pages 767–775, 2002.

20 Subhash Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput.*, 36(4), 2006.

21 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

22 Subhash Khot and Ashok Kumar Ponnuswami. Better inapproximability results for MaxClique, chromatic number and Min-3Lin-Deletion. In *ICALP*, pages 226–237, 2006.

23 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

24 Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. Segmentation problems. *J. ACM*, 51(2):263–280, March 2004.

25 Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *ACM STOC*, 2017. To appear.

26 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense CSPs. In *ICALP*, 2017. To appear.

27 Dana Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. *Theory of Computing*, 11:221–235, 2015.

28 Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: Invariance and optimality. *Ann. Math.*, pages 295–341, 2010.

29 Joseph (Seffi) Naor and Yuval Rabani. Tree packing and approximating k-cuts. In *ACM-SIAM SODA*, pages 26–27, 2001.

30 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

31 René Peeters. The maximum edge biclique problem is NP-complete. *Discrete Appl. Math.*, 131(3):651–654, September 2003.

32 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *ACM STOC*, pages 755–764, 2010.

33 Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between expansion problems. In *IEEE CCC*, pages 64–73, 2012.

34 R. Ravi and Amitabh Sinha. Approximating k-cuts via network strength. In *ACM-SIAM SODA*, pages 621–622, 2002.

35 Huzur Saran and Vijay V. Vazirani. Finding k cuts within twice the optimal. *SIAM J. Comput.*, 24(1):101–108, February 1995.

36 Ola Svensson. Hardness of vertex deletion and project scheduling. *Theory of Computing*, 9(24):759–781, 2013.

37 Mingyu Xiao, Leizhen Cai, and Andrew Chi-Chih Yao. Tight approximation ratio of a general greedy splitting algorithm for the minimum k-way cut problem. *Algorithmica*, 59(4):510–520, 2011.

38 Liang Zhao, Hiroshi Nagamochi, and Toshihide Ibaraki. Approximating the minimum k-way cut in a graph via minimum 3-way cuts. *J. Comb. Optim.*, 5(4):397–410, 2001.

39 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, 2007.

On the Bit Complexity of Sum-of-Squares Proofs*

Prasad Raghavendra[1] and Benjamin Weitz[2]

1 **UC Berkeley, Berkeley, CA, USA**
raghavendra@berkeley.edu
2 **UC Berkeley, Berkeley, CA, USA**
bsweitz@eecs.berkeley.edu

Abstract

It has often been claimed in recent papers that one can find a degree d Sum-of-Squares proof if one exists via the Ellipsoid algorithm. In [16], Ryan O'Donnell notes this widely quoted claim is not necessarily true. He presents an example of a polynomial system with bounded coefficients that admits low-degree proofs of non-negativity, but these proofs necessarily involve numbers with an exponential number of bits, causing the Ellipsoid algorithm to take exponential time. In this paper we obtain both positive and negative results on the bit complexity of SoS proofs.

First, we propose a sufficient condition on a polynomial system that implies a bound on the coefficients in an SoS proof. We demonstrate that this sufficient condition is applicable for common use-cases of the SoS algorithm, such as Max-CSP, Balanced Separator, Max-Clique, Max-Bisection, and Unit-Vector constraints.

On the negative side, O'Donnell asked whether every polynomial system containing Boolean constraints admits proofs of polynomial bit complexity. We answer this question in the negative, giving a counterexample system and non-negative polynomial which has degree two SoS proofs, but no SoS proof with small coefficients until degree $\Omega(\sqrt{n})$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Sum-of-Squares, Combinatorial Optimization, Proof Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.80

1 Introduction

The Sum of squares (SoS) proof system is a versatile and powerful approach to certifying polynomial inequalities. SoS certificates can be shown to underlie a vast number of algorithms in combinatorial optimization. On the one hand, SoS certificates hold the promise of yielding algorithms that possibly refute the notorious unique games conjecture [3, 2, 10]. On the other hand, a flurry of recent works have applied SoS proofs to develop algorithms for problems ranging from constraint satisfaction problems to tensor problems.

To illustrate sum of squares certificates, let us consider the example of the Balanced Separator problem. Here we are given a graph $G = (V, E)$ and the goal is to find a balanced cut $(S, \overline{S})$ with the minimum number of crossing edges. Like many problems in combinatorial optimization, it can be reformulated as a low-degree polynomial optimization problem. Specifically if we associate $\{0, 1\}$ variables $\{x_1, \ldots, x_n\}$ for the vertices of the graph

* This work partially supported by NSF Graduate Research Fellowship (DGE 1106400), NSF Career Award, NSF CCF-1407779 and the Alfred. P. Sloan Fellowship.

G then we can rewrite the Balanced Separator problem as follows:

$$\text{Minimize} \quad \sum_{(i,j)\in E} (x_i - x_j)^2 \quad \text{subject to} \left\{ x_i^2 = x_i \forall i \ , \ \frac{n}{3} \leq \sum_i x_i \leq \frac{2n}{3} \right\} .$$

Here the constraint $x_i^2 = x_i$ ensures $x_i \in \{0,1\}$ while the inequalities enforce the condition that the cut is balanced. More generally, a low-degree polynomial optimization is of the form:

Minimize $r(x)$ subject to
equalities $\mathcal{P} = \{p_i(x) = 0 | i \in [n]\}$ and inequalities $\mathcal{Q} = \{q_i(x) \geq 0 | i \in [m]\}$.

An SoS certificate of a lower bound $r(x) \geq \theta$ is given by a polynomial identity of the form

$$r(x) - \theta = \sum_i h_i(x)^2 + \sum_{i\in[m]} \left(\sum_j^{t_i} s_j^2(x) \right) \cdot q_i(x) + \sum_{i\in[n]} \lambda_i(x) p_i(x) .$$

Notice that for all x satisfying the equalities $\mathcal{P}$ and the inequalities $\mathcal{Q}$, the right hand side of the above identity is manifestly non-negative, thereby certifying that $r(x) \geq \theta$. The degree of the SoS certificate is the maximum degree of the polynomials involved, i.e., $d = \max\{\deg h_i^2, \deg s_j^2 q_i, \deg \lambda_i p_i\}$.

The main appeal of SoS certificates for polynomial optimization is that the existence of a degree d SoS certificate can be formulated as the feasibility of a semidefinite program (SDP). This is the degree d SoS relaxation first introduced by Shor [18], and expanded upon by later works of Nesterov [15], Grigoriev and Vorobjov [9], Lasserre [12, 11] and Parrilo [17]. (see, e.g., [13, 4] for many more details).

The degree d SoS SDP has $n^{O(d)}$ variables, and if the coefficients of p and q are reasonably bounded (smaller than $2^{n^{O(d)}}$), the resulting SDP has a compact description of size $n^{O(d)}$. From this, several works including those by the authors, asserted that the resulting feasibility SDP can be solved in time $n^{O(d)}$ using the Ellipsoid algorithm.

In a recent work, O'Donnell [16] observed that this often repeated claim is far from true. Specifically, O'Donnell exhibited systems of polynomial inequalities with bounded coefficients such that only degree 2 SoS certificates of non-negativity involve coefficients that are doubly exponential in size. Thus all SoS certificates need an exponential number of bits to represent and consequently, the ellipsoid algorithm will incur an exponential running time.

As pointed out by O'Donnell, the issue at hand here is not just that of additive error in the solution, i.e., the difference between testing feasibility and near-feasibility. Indeed, semidefinite programming via the ellipsoid algorithm can only test feasibility up to a very small additive error. However, in a majority of applications of SoS SDP relaxations in combinatorial optimization, the variables in the underlying polynomial system are explicitly bounded (also known as Archimedian). Specifically, these include constraints such as $\{x_i^2 \leq 1 | i \leq [n]\}$, which yield explicit bounds on the values of the variables. In these settings, if there is an approximate SoS certificate for $r(x) \geq \theta$, then there exists a proper SoS certificate for a slightly weaker lower bound $r(x) \geq \theta - o(1)$. Therefore, additive error incurred in semidefinite programming can often be traded off for a slightly weaker objective value. The issue highlighted by O'Donnell is far more serious in that the coefficients of the SoS certificate are too large – thereby directly affecting the runtime of the ellipsoid algorithm.

On a positive note, O'Donnell shows that a polynomial system whose only constraints are the Boolean constraints $\{x_i^2 = x_i | i \in [n]\}$ always admit SoS certificates with polynomial

bit complexity. He proceeds to ask whether all polynomial systems that include Boolean constraints, potentially among others, always admit bounded SoS certificates.

1.1 Our Results

In this work, we further explore the issue of bit complexity of SoS proofs, and obtain both positive and negative results.

First, we present an easily verifiable and broadly applicable set of sufficient conditions under which a polynomial optimization problem has small SoS certificates. Roughly speaking, we show that polynomial systems with *rich* sets of solutions have bounded SoS certificates of non-negativity. Consider a system consisting of polynomial equalities $\mathcal{P}$ and inequalities $\mathcal{Q}$. Our approach consists of looking for assignments S satisfying three criteria (see Definition 5 and Theorem 10 for formal statements).

▶ **Theorem 1.** *Assume $(\mathcal{P}, \mathcal{Q}, S)$ satisfies:*

1. *The assignments S robustly satisfy the inequalities in $\mathcal{Q}$.*
2. *The polynomial calculus (also called Nullstellensatz) proof system is both complete and efficient over S. In other words, all degree d polynomial identities over S can be derived using a degree $O(d)$ polynomial derivation from the equalities $\mathcal{P}$.*
3. *The assignments S are spectrally rich in that smallest non-zero eigenvalue of their covariance matrix is at least $2^{-\operatorname{poly}(n^d)}$.*

Then if r has a degree d proof of non-negativity from $\mathcal{P}$ and $\mathcal{Q}$, it also has a degree $O(d)$ proof of non-negativity with coefficients bounded by $2^{\operatorname{poly}(n^d)}$.

We demonstrate the broad applicability of the above set of sufficient conditions by using them to show upper bounds on bit complexity for Max-CSP, Max-Clique, Matching, Balanced Separator, Max-Bisection, and optimization over the unit sphere. In each case, the above sufficient conditions can be verified easily.

The above set of sufficient conditions are widely applicable in combinatorial optimization, wherein the polynomial system is typically a relaxation of a well-known set of integer solutions. In such a setup with integer solutions, we observe in Section 3 that spectral richness is an immediate consequence of the discrete nature of the set of solutions. Therefore, in all these setups, the only non-trivial thing to verify is the efficiency of the polynomial calculus proof system.

The work of O'Donnell [16] exhibited a polynomial system with bounded coefficients which admitted degree 2 SoS certificate, whose coefficients were necessarily doubly-exponential. However, the variables in this polynomial system were not all Boolean, i.e. did not have the $x_i^2 = x_i$ constraint. In fact, O'Donnell asked whether every polynomial system with Boolean constraints admits a small SoS proof. Moreover, the polynomial system in [16] admits a degree 4 SoS certificate with small bit complexity. This opens up the possibility that one can effectively reduce the bit-complexity by raising the degree of the proof. For instance, if a system admits a degree d SoS certificate then does it always admit a degree 2^d SoS certificate with small bit complexity (even under Boolean constraints)? Unfortunately, we refute both of the above possibilities by exhibiting a counterexample. Formally, we show the following:

▶ **Theorem 2.** *There exists a system of quadratic equations on n variables such that*

- *The system includes the equation $x_i^2 - x_i = 0$ for each $i \in [n]$.*
- *There exists a polynomial with a degree 2 SoS certificate of non-negativity, albeit with doubly exponentially large coefficients.*
- *No SoS certificate of degree $d \leq \sqrt{n}$ has coefficients smaller than $\Omega\left(\frac{1}{n^d} \cdot 2^{\exp(\sqrt{n})}\right)$.*

2 Preliminaries

For a set of real polynomials $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$, we denote their generated ideal in $\mathbb{R}[x]$ by $\langle \mathcal{P} \rangle$ or $\langle p_1, \ldots, p_m \rangle$. We will be working with systems of polynomial constraints, and we will use the $\mathcal{P}$ to denote the equality constraints, and $\mathcal{Q}$ to denote the inequality constraints, i.e. $p(x) = 0$ and $q(x) \geq 0$ for $p \in \mathcal{P}$ and $q \in \mathcal{Q}$. We will usually use S for the set of points satisfying these constraints. We use $\mathbb{R}[x]_d$ for the set of polynomials of degree at most d, and $\mathbb{S}^d_+$ for the cone of positive semidefinite $d \times d$ matrices. We write $\mathbf{v}_d$ for the vector of polynomials whose entries are the elements of the usual monomial basis of $\mathbb{R}[x]_d$. Similarly, we use $\mathbf{v}(\alpha)_d$ for the vector of reals whose entries are the entries of $\mathbf{v}_d$ evaluated at α. We usually omit the dependencies on d as it is clear from context.

2.1 Polynomial Proofs

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a set of polynomials, and let $S = \{x \in \mathbb{R}^n | \forall p \in \mathcal{P} : p(x) = 0\}$. We define a proof of membership in $\langle \mathcal{P} \rangle$ as follows:

▶ **Definition 3.** We say that $r(x)$ has a *derivation* from $\mathcal{P}$ if there is a polynomial identity of the form

$$r(x) = \sum_i^n \lambda_i(x) p_i(x).$$

We say that the proof has degree d if $\max_i \{\deg \lambda_i p_i\} = d$.

A set of polynomials forming a derivation is called a *Polynomial Calculus* (PC) or *Nullstellensatz* proof. The above proof system is useful for proving when polynomials are zero on S, but often we want to prove that they are positive. To that end, let $\mathcal{P} = \{p_1, \ldots, p_n\}$ and $\mathcal{Q} = \{q_1, \ldots, q_m\}$ be two sets of polynomials, and let $S = \{x \in \mathbb{R}^n | \forall p \in \mathcal{P} : p(x) = 0, \forall q \in \mathcal{Q} : q(x) \geq 0\}$. We define a proof of non-negativity as follows:

▶ **Definition 4.** We say that $r(x)$ has a *Sum-of-Squares proof of non-negativity from* $\mathcal{P}$ *and* $\mathcal{Q}$ if there is a polynomial identity of the form

$$r(x) = \sum_i^{t_0} h_i^2(x) + \sum_i^m \left(\sum_j^{t_i} s_j^2(x) \right) q_i(x) + \sum_i^n \lambda_i(x) p_i(x).$$

We say the proof has degree d if $\max\{\deg h_i^2, \deg s_j^2 q_i, \deg \lambda_i p\} = d$.

The idea behind this terminology is that if such a proof exists, then r must be non-negative on S since the first two terms are non-negative, and the last term is zero. We will be concerned with not just the degree of these proofs, but also their bit complexity. To this end, we define the following norms on polynomials and proofs: For $p(x) \in \mathbb{R}[x]$, we write $\|p\|$ for the maximum absolute value of coefficients of p in the standard monomial basis, and for any collection of polynomials $\mathcal{P}$, we write $\|\mathcal{P}\| = \max_{p \in \mathcal{P}} \|p\|$. For a vector $\alpha \in \mathbb{R}^n$, we also write $\|\alpha\|$ for the maximum absolute value of entries of α, and we write $\|S\| = \max_{\alpha \in S} \|\alpha\|$. These norms are usually called *infinity norms* and denoted $\|\cdot\|_\infty$ in other works, but since we do not use other norms in this work we will omit the subscript. Throughout this paper we will assume that the solutions α are *explicitly bounded* by $\|\alpha\| \leq 2^{\text{poly}(n^d)}$.

2.2 Rich Solution Spaces

In this section we define the conditions we require in order to guarantee that SoS proofs from $\mathcal{P}$ and $\mathcal{Q}$ have low bit-complexity. For a polynomial system $(\mathcal{P}, \mathcal{Q})$ and a set $S \subseteq \{x \mid \forall p \in \mathcal{P} : p(x) = 0\}$, define the moment matrix as

$$M_{S,d} \coloneqq E_{\alpha \in S}[\mathbf{v}(\alpha)_d \mathbf{v}(\alpha)_d^T],$$

where the expectation is over the uniform distribution over S. We will omit the subscripts and write M, if S and d are clear from the context.

▶ **Definition 5.** With the above definitions,

- We say that S is δ-*spectrally rich for* $(\mathcal{P}, \mathcal{Q})$ *up to degree* d if every nonzero eigenvalue of $M_{S,d}$ is at least δ.
- We say that $(\mathcal{P}, \mathcal{Q})$ is k-*complete on* S *up to degree* d if every zero eigenvector c of $M_{S,d}$ (which can be seen as a degree d polynomial $c^T \mathbf{v}_d$) has a degree k derivation from $\mathcal{P}$.
- We say that S is ϵ-*robust for* $\mathcal{Q}$ if $\forall q \in \mathcal{Q}, \forall \alpha \in S : q(\alpha) > \epsilon$.

Spectral richness of the solutions S is equivalent to requiring if $p(x)$ is small on S, then there is a polynomial q which agrees with p on S and that has small coefficients. If $(\mathcal{P}, \mathcal{Q}, S)$ satisfies all three conditions then we say that S is (δ, k, ϵ)-*rich* for $(\mathcal{P}, \mathcal{Q})$ up to degree d. If $1/\delta = 2^{\mathrm{poly}(n^d)}$, $k = O(d)$, and $1/\epsilon = 2^{\mathrm{poly}(n^d)}$ we simply say S is rich for $(\mathcal{P}, \mathcal{Q})$ or simply rich. We choose these bounds because Theorem 10 will imply that any constraints with a rich solution space has proofs of non-negativity that can be taken to have polynomial bit complexity.

▶ Remark. There is nothing special about the uniform distribution on S for these definitions. In fact, our results hold if there is *any* distribution over a set $S \subseteq \{x \mid \forall p \in \mathcal{P} : p(x) = 0\}$ with the above properties. In this work we consider mostly combinatorial problems where S is finite, and the uniform distribution is sufficient for all of our examples, so we restrict to this case for simplicity.

Before we get into the proof of the main theorem, we exhibit polynomial systems that admit rich solutions.

3 Examples with Rich Solution Spaces

In this section we present examples of polynomial systems that admit rich solution spaces. First, we consider the case $S \subseteq \{0,1\}^n$. In this case, the spectral richness is a consequence of the following easy observation.

▶ **Lemma 6.** *Let $M \in \mathbb{S}_+^N$ be an integer matrix with $|M_{ij}| \leq B$ for all $i, j \in [N]$. The smallest non-zero eigenvalue of M is at least $(BN)^{-N}$.*

Proof. Let A be a full-rank principal minor of M (which must exist because M is PSD and has a Cholesky decomposition), and for convenience let it be at the upper-left block of M (by permuting rows and columns if necessary). We claim the least eigenvalue of A lower bounds the least nonzero eigenvalue of M. Since M is symmetric, there must be a C such that

$$M = \begin{bmatrix} I \\ C \end{bmatrix} A \begin{bmatrix} I & C^T \end{bmatrix}.$$

Let $P = [I, C^T]$, ρ be the least eigenvalue of A, and x be a unit vector perpendicular to the zero eigenspace of M. Then we have $x^T M x = (Px)^T A (Px) \geq \rho x^T P^T P x$. Now $P^T P$ has

the same nonzero eigenvalues as $PP^T = I + C^T C \succeq I$, and the zero eigenspace of $P^T P$ is the same as the zero eigenspace of M. Because x is perpendicular to the zero eigenspace, $x^T P^T P x \geq 1$, and so every nonzero eigenvalue of M is at least ρ. Now A is a full-rank bounded integer matrix with dimension at most N. The magnitude of its determinant is at least 1 and all eigenvalues are at most $N \cdot B$. Therefore, its least eigenvalue must be at least $(BN)^{-N}$ in magnitude. ◀

▶ **Lemma 7.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be such that $S \subseteq \{0,1\}^n$. Then S is δ-spectrally rich with $\frac{1}{\delta} = 2^{\text{poly}(n^d)}$.*

Proof. Recall $M = E_{\alpha \in S}[\mathbf{v}(\alpha)\mathbf{v}(\alpha)^T]$, and note that $|S| \cdot M$ is an integer $O(n^d) \times O(n^d)$ matrix with entries at most 2^n. The proof follows by applying Lemma 6. ◀

To prove completeness, we typically want to show two things. First, that every degree d polynomial in $\langle \mathcal{P} \rangle$ has a degree at most k derivation. Second, that there are no polynomials outside $\langle \mathcal{P} \rangle$ that are zero on S. This second condition can be thought of as saying that the set of equations $\mathcal{P}$ is somehow maximal, i.e., if there are extra polynomial equalities implied by $\mathcal{Q}$, they should be included in $\mathcal{P}$. Here we consider a few examples.

Max-CSP: $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\}$

Here $S = \{0,1\}^n$. Any polynomial p of degree d can be multilinearized one monomial at a time. Specifically, we can find degree d multilinear p^* such that $p - p^* = 0$ has a degree d derivation from $\mathcal{P}$. Furthermore, the multilinear polynomial p^* is zero over S if and only if all its coefficients are zero. Thus $\mathcal{P}$ is d-complete up to degree d for all $d \in \mathbb{N}$.

Max-Clique: $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\} \cup \{x_i x_j | (i,j) \notin E\}$

Here S is the set of all cliques in the graph. Suppose p is a polynomial that is identically zero over S. Without loss of generality, we may assume p is multilinear, if otherwise we can multilinearize it using $\{x_i^2 - x_i | i \in [n]\}$. For a multilinear polynomial $p(x) = \sum_{\alpha \subset [n]} \hat{p}_\alpha x_\alpha$, we claim that if $p(x) = 0 \forall x \in S$ then for all cliques $\alpha \subset [n]$, the corresponding coefficient $\hat{p}_\alpha = 0$, i.e., all non-zero coefficients of p are non-cliques. Suppose not, then let α be the smallest clique with $\hat{p}_\alpha \neq 0$. Then, we will have $p(\mathbb{I}_\alpha) = \hat{p}_\alpha \neq 0$, a contradiction. Since all coefficients of p are non-cliques, each monomial in p can be eliminated using an appropriate polynomial from $\{x_i x_j | (i,j) \notin E\}$.

▶ Remark. More generally, the above two cases are special cases of the following general setup: $\mathcal{Q}$ is empty, and $\mathcal{P}$ is a Gröbner basis. A Gröbner basis for an ideal is a generating set of polynomials that allow a well-defined multivariate polynomial division (see [1] for more information). Computing the Gröbner basis is often the first step in practical polynomial equation solvers, and we note the following easy lemma:

▶ Lemma 8. *If $\mathcal{Q} = \emptyset$ and $\mathcal{P}$ is a Gröbner basis for $\langle \mathcal{P} \rangle$, then S is d-complete up to degree d.*

Proof. If $\mathcal{P}$ is a Gröbner basis, then every degree d polynomial in $\langle \mathcal{P} \rangle$ has a degree d derivation via multivariate division. Because $\mathcal{Q} = \emptyset$, the polynomials that are zero on S are exactly the polynomials in $\langle \mathcal{P} \rangle$. ◀

Balanced Separator: $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\}$, $\mathcal{Q} = \{2n/3 - \sum_i x_i, \sum_i x_i - n/3\}$

The solution space S here is all bit strings with hamming weight between $n/3$ and $2n/3$. Suppose r is a polynomial that is zero on S. Without loss of generality, we may assume that r is multilinear by using the constraints $\{x_i^2 - x_i | i \in [n]\}$. Suppose r is a non-zero multilinear polynomial which is zero on S, then its symmetrized version $r^* = \frac{1}{n!}\sum_{\sigma \in \mathcal{S}_n} \sigma r$ must also be zero on S, where σ acts by permuting the variable names. However, r^* is a univariate polynomial in $\sum_i x_i$ (modulo the Boolean constraints). This univariate polynomial has $n/3$ zeros, and thus must have degree at least $n/3$. Since symmetrizing does not change degree, we conclude that r also has degree at least $n/3$. Thus every non-zero multilinear polynomial that is zero on S but not in $\langle \mathcal{P} \rangle$, has degree at least $n/3$. Therefore the system is d-complete up to degree d for $d \leq \frac{n}{3}$. The polynomials in $\mathcal{Q}$ can be perturbed by $1/2$ to make them $1/2$-robust, and thus S is rich for $(\mathcal{P}, \mathcal{Q})$.

Matching: $\mathcal{P} = \{x_{ij}^2 - x_{ij} | i, j \in [n]\} \cup \{\sum_i x_{ij} - 1 | i \in [n]\} \cup \{x_{ij}x_{ik} | i, j, k \in [n]\}$

These constraints are $2d$-complete as proven in [5].

Max-Bisection: $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\} \cup \{\sum_i x_i - n/2\}$

We will prove in Section 6 that these constraints are d-complete. The proof will be very similar to the one for MATCHING, due to the similar symmetry of the constraints.

Unit-Vector: $\mathcal{P} = \{\sum_i x_i^2 - 1\}$

Here $S = \{x : \|x\| = 1\}$. This constraint appears frequently in tensor norm problems as a way to enforce scaling. Since $\mathcal{Q} = \emptyset$, it is clearly robust. It may be well-known that $\mathcal{P}$ is d-complete, but we could not find a reference so we record it here for completeness. Let $p(x)$ be any degree d polynomial which is zero on the unit sphere, and define $p_0(x) = p(x) + p(-x)$. Clearly p_0 is also zero on the unit sphere, with degree $k = 2\lfloor (d+1)/2 \rfloor$. Note that p_0 has only terms of even degree. Define a sequence of polynomials $\{p_i\}_{i \in \{0,\ldots,k\}}$ as follows. Define q_i to be the part of p_i which has degree strictly less than k, and let $p_{i+1} = p_i + q_i \cdot (\sum_i x_i^2 - 1)$. Then each p_i is zero on the unit sphere and has no monomials of degree strictly less than $2i$. Thus $p_{k/2}$ is homogeneous of degree k. But then $p(tx) = t^k p_k(x) = 0$ for any unit vector x and $t > 0$, and thus $p_k(x)$ must be the zero polynomial. This implies that p_0 is a multiple of $\sum_i x_i^2 - 1$. The same logic shows that $p(x) - p(-x)$ is also a multiple of $\sum_i x_i^2 - 1$, and thus so is $p(x)$. Now $\langle \mathcal{P} \rangle$ is principal, so every degree d polynomial in it has a degree d derivation, so $(\mathcal{P}, \mathcal{Q}, S)$ is d-complete.

To prove spectral-richness, we note that in [7] the author gives an exact formula for each entry of the matrix $M = \int_S p(x)$ for any polynomial p. The formulas imply that $(n+d)!\pi^{-n/2}M$ is an integer matrix with entries (very loosely) bounded by $(n+d)!d!2^n$. By Lemma 6, we conclude that S is δ-spectrally rich with $1/\delta = 2^{\mathrm{poly}(n^d)}$.

We collect the examples discussed in this section here:

▶ **Corollary 9.** *The following constraints admit rich solutions:*

- MAX-CSP*:* $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\}$.
- MAX-CLIQUE*:* $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\} \cup \{x_i x_j | (i,j) \notin E\}$.
- BALANCED SEPARATOR*:* $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\}$, $\mathcal{Q} = \{2n/3 - \sum_i x_i, \sum_i x_i - n/3\}$.
- MATCHING*:* $\mathcal{P} = \{x_{ij}^2 - x_{ij} | i, j \in [n]\} \cup \{\sum_i x_{ij} - 1 | i \in [n]\} \cup \{x_{ij}x_{ik} | i, j, k \in [n]\}$.
- MAX-BISECTION*:* $\mathcal{P} = \{x_i^2 - x_i | i \in [n]\} \cup \{\sum_i x_i - n/2\}$.
- UNIT-VECTOR*:* $\mathcal{P} = \{\sum_i x_i^2 - 1\}$.

3.1 Limitations

While Theorem 10 allows us to prove that many different systems of polynomial constraints have well-behaved SoS proofs, there are a few areas where it comes up short. Most noticeably, to contain a rich set of solutions the solution space has to be nonempty. This can be a problem when trying to find SoS proofs of infeasibility. For example, one common technique is to introduce lower bounds on an objective function $f(x)$ of a maximization problem as constraints and attempt to use SoS to find a refutation, i.e. a proof of non-negativity for the constant polynomial -1. We are unable to show that these proofs can be taken to have polynomial bit complexity since they have empty solution spaces. As another example, we are unable to use our framework to show that refutations of the knapsack constraints use only polynomially many bits, even though it is clear by simply examining these known refutations that they only involve small coefficients.

4 Rich Solution Spaces Yield Bounded SoS Proofs

In this section we prove our main theorem:

▶ **Theorem 10.** *Let $\mathcal{P} = \{p_1, \ldots, p_m\}$ and $\mathcal{Q} = \{q_1, \ldots, q_\ell\}$ be sets of polynomials with $S \subseteq \{\alpha \in \mathbb{R}^n | \forall p \in \mathcal{P} : p(\alpha) = 0\}$. Assume that the set S is (k, δ, ϵ)-rich for $(\mathcal{P}, \mathcal{Q})$.*

Let $r(x)$ be a polynomial non-negative on S, and assume r has a degree d sum-of-squares proof of non-negativity

$$r(x) = \sum_{i=1}^{t_0} h_i^2 + \sum_{i=1}^{\ell} \left(\sum_{j=1}^{t_i} s_j^2 \right) q_i + \sum_{i=1}^{m} \lambda_i p_i.$$

Then r has a degree k sum-of-squares proof of non-negativity such that the coefficients of every polynomial appearing in the proof are bounded by $2^{\mathrm{poly}(n^k, \log \frac{1}{\delta}, \log \frac{1}{\epsilon})}$. In particular, if S is rich then every coefficient can be written down with only $\mathrm{poly}(n^d)$ bits.

Proof. First, we rewrite the proof into a more convenient form before proving bounds on each individual term. Because the elements of $\mathbf{v}$ are a basis for $\mathbb{R}[x]_d$, every polynomial in the proof can be expressed as $c^T \mathbf{v}$, where c is a vector of reals:

$$\begin{aligned} r(x) &= \sum_{i=1}^{t_0} (c_i^T \mathbf{v})^2 + \sum_{i=1}^{\ell} \left(\sum_{j=1}^{t_i} (d_{ij}^T \mathbf{v})^2 \right) q_i + \sum_{i=1}^{m} \lambda_i p_i \\ &= \langle C, \mathbf{v}\mathbf{v}^T \rangle + \sum_{i=1}^{\ell} \langle D_i, \mathbf{v}\mathbf{v}^T \rangle q_i + \sum_{i=1}^{m} \lambda_i p_i \,. \end{aligned}$$

for PSD matrices C, $D_1, \ldots, D_\ell$. Next, we average this polynomial identity over all the points $\alpha \in S$:

$$\mathop{\mathbb{E}}_{\alpha \in S}[r(\alpha)] = \langle C, \mathop{\mathbb{E}}_{\alpha \in S}[\mathbf{v}(\alpha)\mathbf{v}(\alpha)^T] \rangle + \sum_{i=1}^{\ell} \langle D_i, \mathop{\mathbb{E}}_{\alpha \in S}[q_i(\alpha)\mathbf{v}(\alpha)\mathbf{v}(\alpha)^T] \rangle + 0 \,.$$

The LHS is at most $\mathrm{poly}(\|r\|, \|S\|)$, and the RHS is a sum of positive numbers, so the LHS is a bound on each term of the RHS. We would like to say that since S is δ-spectrally rich, the first term is at least $\delta \operatorname{Tr}(C)$. Unfortunately the averaged matrix may have zero eigenvectors, and it is possible that C could have very large eigenvalues in these directions. However

these eigenvectors must correspond to polynomials that are zero on S. Because $(\mathcal{P}, \mathcal{Q}, S)$ is complete, these can be absorbed into the final term. More formally, let $\Pi = \sum_u uu^T$ be the projector onto the zero eigenspace of $M = \mathbb{E}_{\alpha \in S}[\mathbf{v}(\alpha)\mathbf{v}(\alpha)^T]$. Because $(\mathcal{P}, \mathcal{Q}, S)$ is complete, for each u we have a degree k derivation $u^T\mathbf{v} = \sum_i \sigma_{ui} p_i$. Then $\Pi\mathbf{v}\mathbf{v}^T = \sum_u (u^T\mathbf{v})u\mathbf{v}^T$. Thus we can write

$$\begin{aligned}\langle C, \mathbf{v}\mathbf{v}^T\rangle &= \langle C, (\Pi + \Pi^\perp)\mathbf{v}\mathbf{v}^T(\Pi + \Pi^\perp)\rangle \\ &= \langle C, \Pi^\perp \mathbf{v}\mathbf{v}^T \Pi^\perp\rangle + \sum_u u^T\mathbf{v}\left(\langle C, \Pi^\perp \mathbf{v}u^T + \mathbf{v}u^T\Pi^\perp + \mathbf{v}u^T\Pi\rangle\right) \\ &= \langle \Pi^\perp C \Pi^\perp, \mathbf{v}\mathbf{v}^T\rangle + \sum_i \sigma_i p_i.\end{aligned}$$

Doing the same for the other terms and setting $C' = \Pi^\perp C \Pi^\perp$ and similarly for D'_i, we get a new proof:

$$r(x) = \langle C', \mathbf{v}\mathbf{v}^T\rangle + \sum_{i=1}^{\ell} \langle D'_i, \mathbf{v}\mathbf{v}^T\rangle q_i + \sum_{i=1}^{m} \lambda'_i p_i.$$

Now after averaging over S, the zero eigenspaces of C' and each D'_i are contained in the zero eigenspace of M. Furthermore, ϵ-robustness implies, for each i,

$$\langle D'_i, \mathop{\mathbb{E}}_{\alpha \in S}[\mathbf{v}(\alpha)\mathbf{v}(\alpha)^T q_i(\alpha)]\rangle \geq \epsilon \langle D'_i, \mathop{\mathbb{E}}_{\alpha \in S}[\mathbf{v}(\alpha)\mathbf{v}(\alpha)^T]\rangle .$$

Taken with the δ-spectral richness, we have

$$\text{poly}(\|r\|, \|S\|) \geq \delta \operatorname{Tr}(C') + \sum_{i=1}^{\ell} \epsilon\delta \operatorname{Tr}(D'_i).$$

The Frobenius norm of any PSD matrix is bounded by its trace, so we conclude that C' and each D'_i have entries bounded by $\text{poly}(\|r\|, \|S\|, \frac{1}{\delta}, \frac{1}{\epsilon})$.

The only thing left to do is to bound the coefficients λ'_i, but this is easy because the SoS proof is linear in these coefficients. If we imagine the coefficients of the λ'_i as variables, then the linear system induced by the polynomial identity

$$r(x) - \langle C', \mathbf{v}\mathbf{v}^T\rangle - \sum_{i=1}^{\ell} \langle D'_i, \mathbf{v}\mathbf{v}^T\rangle = \sum_{i=1}^{m} \lambda'_i p_i$$

is clearly feasible, and the coefficients of the LHS are bounded by $\text{poly}(\|r\|, \|S\|, \frac{1}{\delta}, \frac{1}{\epsilon})$. There are $O(n^k)$ variables, so by Cramer's rule, the coefficients of the λ'_i can be taken to be bounded by $\text{poly}(\|\mathcal{P}\|^{n^k}, \frac{1}{\delta}, \frac{1}{\epsilon}, \|r\|, \|S\|, n!)$. $\|\mathcal{P}\|, \|r\| \leq 2^{\text{poly}(n^d)}$ as they are considered part of the input, $\|S\| \leq 2^{\text{poly}(n^d)}$ by the explicitly bounded assumption, and $d \leq k$. Thus, this bound is at most $2^{\text{poly}(n^k, \log\frac{1}{\delta}, \log\frac{1}{\epsilon})}$. ◀

5 Boolean Systems With No Small-Coefficient Proofs

In [16], the author gives an example of a polynomial system for which degree two SoS proofs can certify non-negativity of a certain polynomial, but the proofs necessarily involves coefficients of doubly-exponential size. However, there are two weaknesses in his example system. First, it is not a Boolean one, i.e. it contains variables y_i for which the constraint $y_i^2 - y_i = 0$ is not present in the constraints. Many practical optimization problems have

Boolean constraints, and in [16], the author hoped that having those constraints might suffice to imply that all proofs could have small bit complexity. Second, while the degree two proofs must have exponential bit complexity, there were degree four proofs of non-negativity with polynomial bit complexity. In this section, we strengthen his counterexample, giving an example of a Boolean system with n variables for which there is a polynomial that has a degree two proof of non-negativity, but no proof with polynomial bit complexity until degree $\Omega(\sqrt{n})$.

5.1 A First Example

The original example given in [16] essentially contains the following system whose repeated squaring is responsible for the blowup of the coefficients in the proofs:

$$y_1^2 - y_2 = 0, \quad y_2^2 - y_3 = 0, \quad \ldots, \quad y_{n-1}^2 - y_n = 0, \quad y_n^2 = 0.$$

Clearly, the only solution to the system is $(0, 0, 0, \ldots, 0)$, and therefore the polynomial $\epsilon - y_1$ must be non-negative over the solution space for any $\epsilon > 0$. It is not as obvious whether or not an SoS proof of this non-negativity exists. It turns out that there is a degree two SoS proof as follows:

$$\epsilon - y_1 \equiv \left(\sqrt{\frac{\epsilon}{n}} - \left(\frac{n}{4\epsilon}\right)^{1/2} y_1\right)^2 + \left(\sqrt{\frac{\epsilon}{n}} - \left(\frac{n}{4\epsilon}\right)^{3/2} y_2\right)^2 + \left(\sqrt{\frac{\epsilon}{n}} - \left(\frac{n}{4\epsilon}\right)^{7/2} y_3\right)^2 + \\ + \cdots + \left(\sqrt{\frac{\epsilon}{n}} - \left(\frac{n}{4\epsilon}\right)^{(2^n-1)/2} y_n\right)^2. \qquad (*)$$

where the $\equiv$ is equality modulo the ideal generated by the constraints. Of course, this proof involves coefficients of doubly-exponential size, but one can prove that they are required. We will take $\epsilon < 1/2$ for simplicity. We will define a linear functional $\phi : \mathbb{R}[Y]_d \to \mathbb{R}$ satisfying the following:

- $\phi[\epsilon - y_1] = -\epsilon$
- $\phi[p^2] \geq 0$ for any p^2 of degree at most d
- $\phi[\sigma_i(y_i^2 - y_{i+1})] = 0$ for any $i \leq n-1$ and σ_i of degree at most $d-2$
- $|\phi[\lambda y_n^2]| \leq (2\epsilon)^{2^{n-1}} n^d \|\lambda\|$.

If such a ϕ exists, then for any degree d SoS proof of non-negativity

$$\epsilon - y_1 = \sum_i h_i(y)^2 + \sum_{i=1}^{n-1} \sigma_i(y_i^2 - y_{i+1}) + \lambda \cdot y_n^2,$$

apply ϕ to both sides. We obtain $-\epsilon \leq P + 0 + \phi[\lambda y_n^2]$, where $P \geq 0$. Because $|\phi[\lambda y_n^2]| \leq (2\epsilon)^{2^{n-1}} n^d \|\lambda\|$, λ must contain a coefficient of size at least $\Omega(\frac{1}{n^d}\left(\frac{1}{2\epsilon}\right)^{2^n})$.

To show that such a ϕ exists, we define it as follows. By the constraints, every monomial is equivalent to some power of y_1. For example, $y_1 y_2 y_3 \equiv y_1^7$. More generally, the constraints imply that $\prod_{i=1}^n y_i^{\beta_i} = y_1^{\sum_{j=1}^n 2^{j-1}\beta_j}$. Define ϕ by,

$$\phi\left(\prod_{i=1}^n y_i^{\beta_i}\right) = (2\epsilon)^{\sum_i 2^{i-1}\beta_i}.$$

One can easily check that this ϕ satisfies the above. Note that none of the variables y_i in the above system are Boolean, which we achieve in the upcoming section.

5.2 A Boolean System

One simple way to try to make the system Boolean is to just add the constraints $y_i^2 = y_i$ to the system. Unfortunately, in that case it is easy to prove that $y_i - y_j = 0$ for each i and j, and of course $y_n = y_n^2 = 0$. It is too easy for SoS to figure out what each y_i should look like. Previously, the variables were unconstrained in any way, and we want to imitate that. We draw inspiration from the Knapsack problem, and we instead replace each instance of the variable y_i with a sum of $2k$ Boolean variables

$$y_i \to \sum_j w_{ij} - k,$$

and we consider the non-negative polynomial $\epsilon - (\sum_j w_{1j} - k)$. Clearly there is a degree two proof of non-negativity for this polynomial since we can just replace each instance of y_i with $\sum_j w_{ij} - k$ in $(*)$.

It remains to show that there are no other proofs that have only small coefficients. Here, we use the fact that the Knapsack problem is hard for SoS: there is no SoS proof of degree less than $\Omega(k)$ that $\sum_j w_{ij} - k$ is not equal to any number $r \in (0, 1)$ [8]. This allows us to use the Knapsack pseudodistribution to "pretend" that $\sum_j w_{ij} - k = (2\epsilon)^{2^{i-1}}$. Specifically, for each $r \in (0, 1)$, there is a linear functional ϕ_r defined on polynomials of $2k$ Boolean variables which satisfies

- $\phi_r[\sigma_{ij}(w_{ij}^2 - w_{ij})] = 0$ for any σ_{ij} up to degree $O(k)$
- $\phi_r[\lambda \cdot ((\sum_j w_{ij} - k) - r)] = 0$ for any polynomial λ up to degree $O(k)$
- $\phi_r[p^2] \geq 0$ for any polynomial p^2 of degree at most $O(k)$.

Now, take the linear functional Φ defined on each polynomials of $2kn$ variables defined in the following way: Let $T = T_1 \cup T_2 \cup \cdots \cup T_n$ where T_i is a multiset that contains only the variables corresponding to y_i, and let w_T denote the associated monomial. Then define

$$\Phi[w_T] = \phi_{2\epsilon}(w_{T_1})\phi_{(2\epsilon)^2}(w_{T_2}) \ldots \phi_{(2\epsilon)^{2^{n-1}}}(w_{T_n}).$$

Clearly Φ is non-negative on squares and $\Phi[\sigma_{ij}(w_{ij}^2 - w_{ij})] = 0$ for any σ_{ij} up to degree $\Omega(k)$. Because $\Phi[\lambda(\sum_j w_{ij} - k)] = \Phi[(2\epsilon)^{2^{i-1}}\lambda]$, Φ also satisfies $\Phi[\lambda((\sum_j w_{ij} - k)^2 - (\sum_j w_{i+1,j} - k))] = 0$ for each λ and $1 \leq i \leq n-1$. Finally, because each variable is Boolean, Φ of any monomial is at most one, so for any monomial w_M, $\Phi[w_M(\sum_j w_{nj} - k)^2] = \Phi[(2\epsilon)^{2^{n-1}} w_M] \leq (2\epsilon)^{2^{n-1}}$. There are at most $(nk)^d$ monomials, so $\Phi[\lambda(\sum_j w_{nj} - k)^2] \leq (nk)^d (2\epsilon)^{2^{n-1}} \|\lambda\|$. Just as before, the existence of Φ implies that any degree d proof of non-negativity for $\epsilon - (\sum_j w_{1j} - k)$ must contain coefficients of size at least $\Omega(\frac{1}{(nk)^d} \cdot \left(\frac{1}{2\epsilon}\right)^{2^n})$. If we set $k = n$, then there are n^2 variables and no proof of non-negativity with coefficients smaller than doubly-exponential until degree n. This proves Theorem 2.

6 Max-Bisection Constraints

In this section, we prove our earlier claim that the Max-Bisection constraints admit rich solutions. Recall the constraints:

$$\mathcal{P}(n) = \left\{x_i^2 - x_i | i \in [2n]\right\} \cup \left\{\sum_i x_i - n\right\}.$$

Recall that to prove S is rich, we have to prove that it is spectrally rich, robust, and complete. Since the solution space lies in the hypercube, it is spectrally rich by Lemma 7, and it is clearly robust since $\mathcal{Q}$ is empty. It remains to prove that it is complete for some k. This proof follows a very similar path to [5], due to the similar symmetry of the constraints.

▶ **Lemma 11.** *$\mathcal{P}(n)$ is d-complete for any $d \leq n$.*

▶ Remark. A reviewer has pointed out that this is already essentially known by combining Corollary B.6 of [14] with Theorem 3.5 of [6]. We include a proof here for completeness.

Proof. Let $S(n)$ denote the solution space of $\mathcal{P}(n)$, and let $M = \mathbb{E}_{\alpha \in S}[\mathbf{v}(\alpha)\mathbf{v}(\alpha)^T]$. Any zero eigenvector c of M can be associated with a polynomial $c^T\mathbf{v}$. Since $c^T M c = E_{\alpha \in S}[(c^T v(\alpha))^2]$ and $c^T M c = 0$, we must have $c^T v(\alpha) = 0$ for each $\alpha \in S$. We argue that any degree d polynomial which is identically zero on $S(n)$ must have a degree d derivation from $\mathcal{P}(n)$.

We proceed by induction on d. If $d = 0$, the only constant polynomial zero on $S(n)$ is the zero polynomial, which has the trivial derivation. Now consider the case of $d = c + 1$. We proceed in two parts. First, if r is fully symmetric, we show that it has a degree d derivation. Secondly, for any polynomial p which is zero on $S(n)$, we prove that $p - \frac{1}{(2n)!}\sum_{\sigma \in \mathcal{S}_n} \sigma p$ has a degree d derivation from $\mathcal{P}$, where σ acts on p by permuting the labels of the variables. Taken together, these two facts imply that r has a degree d derivation from $\mathcal{P}(n)$.

To prove the first part, note that a symmetric polynomial r is a linear combination of the elementary symmetric polynomials $e_1, \ldots, e_c$, and it is clear that $e_k(x)$ can be derived by taking the polynomial $(\sum_i x_i - n)^k$, reducing it to multilinear using the Boolean constraints, and then reducing by $e_l(x)$ for each $l < k$. This will result in a constant polynomial, which must be the zero polynomial since we are only adding polynomials which are zero on $S(n)$, so the resulting polynomial must be zero on $S(n)$.

To prove the second part, let σ_{ij} be the transposition of labels i and j, and consider the polynomial $r - \sigma_{ij} r$. Writing $r = r_i x_i + r_j x_j + r_{ij} x_i x_j + q_{ij}$, where none of r_i,r_j,r_{ij}, nor q_{ij} depend on x_i or x_j, we can rewrite

$$r - \sigma_{ij} r = (r_i - r_j)(x_i - x_j).$$

Now because $r - \sigma_{ij} r$ evaluates to zero on any Boolean string with exactly n ones, if we set $x_i = 1$ and $x_j = 0$, we know that $r_i - r_j$ is a polynomial that must evaluate to zero on any Boolean string with exactly $n - 1$ ones. Because $\deg(r_i - r_j) = d - 1$, by the inductive hypothesis, $r_i - r_j$ has a degree $d-1$ proof from $\mathcal{P}(n-1)$ (since $d \leq n$, clearly $d - 1 \leq n - 1$). This implies that $(r_i - r_j)(x_i - x_j)$ has a degree $d - 1$ proof from $\mathcal{P}(n)$:

$$\begin{aligned}
(r_i - r_j)(x_i - x_j) &= \left[\sum_{t \neq i,j} \lambda_t (x_t^2 - x_t) + \lambda \left(\sum_{t \neq i,j} x_t - (n-1)\right)\right](x_i - x_j) \\
&= \sum_t \lambda'_t (x_t^2 - x_t) + \lambda \left(\sum_{t \neq i,j} x_t - (n-1) + (x_i + x_j - 1)\right)(x_i - x_j) \\
&= \sum_t \lambda'_t (x_t^2 - x_t) + \lambda' \left(\sum_t x_t - n\right)
\end{aligned}$$

where we used the fact that $(x_i + x_j - 1)(x_i - x_j) - (x_i^2 - x_i) + (x_j^2 - x_j) = 0$. The degree of this derivation is at most d because each λ_t has degree at most $d - 3$, and $\lambda'_t = \lambda_t(x_i - x_j)$, and similarly for λ. Thus the inductive hypothesis implies that $r - \sigma_{ij} r$ has a degree d derivation, and since transpositions generate the symmetric group, this implies that $r - \frac{1}{(2n)!}\sum_{\sigma \in \mathcal{S}_n} \sigma r$ has a degree d proof from $\mathcal{P}(n)$. ◀

▶ Remark. In this example, $\mathcal{P}$ is not a Gröbner basis for its ideal $\langle \mathcal{P} \rangle$. Indeed, the Gröbner basis for this ideal has exponential size. This is an example where our framework is applicable, even though Gröbner bases are intractable to compute.

References

1 William Adams and Philippe Loustaunau. *An Introduction to Gröbner Bases*. American Mathematical Society, 1994.

2 B. Barak, P. Raghavendra, and D. Steurer. Rounding semidefinite programming hierarchies via global correlation. In *Proc. FOCS*, pages 472–481. IEEE, 2011.

3 Boaz Barak, Fernando G. S. L. Brandao, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC'12, pages 307–326, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214006.

4 Boaz Barak and David Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. *In Proceedings of the 2014 International Congress of Mathematicians. International Mathematical Union*, 2014.

5 Gábor Braun, Jonah Brown-Cohen, Arefin Huq, Sebastian Pokutta, Prasad Raghavendra, Aurko Roy, Benjamin Weitz, and Daniel Zink. The Matching Problem Has No Small Symmetric SDP. In *Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'16)*, pages 1067–1078. SIAM, 2016. URL: http://dl.acm.org/citation.cfm?id=2884435.2884510.

6 Yuval Filmus and Elchanan Mossel. Harmonicity and invariance on slices of the boolean cube. In *Proc. of the 31st Conf. on Computational Complexity (CCC'16)*, LIPIcs, pages 16:1–16:13, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2016.16.

7 Gerald B. Folland. How to integrate a polynomial over a sphere. *The American Mathematical Monthly*, 108(5):446–448, 2001. URL: http://www.jstor.org/stable/2695802.

8 D. Grigoriev. Complexity of Positivstellensatz proofs for the knapsack. *computational complexity*, 10(2):139–154, 2001. doi:10.1007/s00037-001-8192-0.

9 Dima Grigoriev and Nicolai Vorobjov. Complexity of Null-and Positivstellensatz proofs. *Annals of Pure and Applied Logic*, 113(1-3):153–160, 2001.

10 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 482–491, 2011. doi:10.1109/FOCS.2011.36.

11 Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

12 Jean Bernard Lasserre. Optimisation globale et théorie des moments. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 331(11):929–934, 2000.

13 Monique Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

14 Troy Lee, Anupam Prakash, Ronald de Wolf, and Henry Yuen. On the sum-of-squares degree of symmetric quadratic functions. In *Proc. of the 31st Conf. on Computational Complexity (CCC'16)*, LIPIcs, pages 17:1–17:31, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2016.17.

15 Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, pages 405–440. Springer, 2000.

16 Ryan O'Donnell. SOS is not obviously automatizable, even approximately. *Innovations in Theoretical Computer Science (ITCS)*, 2017.

17 Pablo A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

18 Naum Z. Shor. Class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6):731–734, 1987.

The Dependent Doors Problem: An Investigation into Sequential Decisions without Feedback*†

Amos Korman[1] and Yoav Rodeh[2]

1 **CNRS and University Paris Diderot, Paris, France**
amos.korman@irif.fr

2 **Weizmann Institute of Science, Rehovot, Israel**
yoav.rodeh@gmail.com

Abstract

We introduce the *dependent doors problem* as an abstraction for situations in which one must perform a sequence of possibly dependent decisions, without receiving feedback information on the effectiveness of previously made actions. Informally, the problem considers a set of d doors that are initially closed, and the aim is to open all of them as fast as possible. To open a door, the algorithm knocks on it and it might open or not according to some probability distribution. This distribution may depend on which other doors are currently open, as well as on which other doors were open during each of the previous knocks on that door. The algorithm aims to minimize the expected time until all doors open. Crucially, it must act at any time without knowing whether or which other doors have already opened. In this work, we focus on scenarios where dependencies between doors are both positively correlated and acyclic.

The fundamental distribution of a door describes the probability it opens in the best of conditions (with respect to other doors being open or closed). We show that if in two configurations of d doors corresponding doors share the same fundamental distribution, then these configurations have the same optimal running time up to a universal constant, no matter what are the dependencies between doors and what are the distributions. We also identify algorithms that are optimal up to a universal constant factor. For the case in which all doors share the same fundamental distribution we additionally provide a simpler algorithm, and a formula to calculate its running time. We furthermore analyse the price of lacking feedback for several configurations governed by standard fundamental distributions. In particular, we show that the price is logarithmic in d for memoryless doors, but can potentially grow to be linear in d for other distributions.

We then turn our attention to investigate precise bounds. Even for the case of two doors, identifying the optimal sequence is an intriguing combinatorial question. Here, we study the case of two cascading memoryless doors. That is, the first door opens on each knock independently with probability p_1. The second door can only open if the first door is open, in which case it will open on each knock independently with probability p_2. We solve this problem almost completely by identifying algorithms that are optimal up to an additive term of 1.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.2.2 Sequencing and Scheduling

Keywords and phrases No Feedback, Sequential Decisions, Probabilistic Environment, Exploration and Exploitation, Golden Ratio

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.81

* The full version of this paper appears in https://arxiv.org/abs/1704.06096.

† This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 648032).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 81; pp. 81:1–81:13

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

1 Introduction

Often it is the case that one must accomplish multiple tasks whose success probabilities are dependent on each other. In many cases, failure to achieve one task will tend to have a more negative affect on the success probabilities of other tasks. In general, such dependencies may be quite complex, and balancing the work load between different tasks becomes a computational challenge. The situation is further complicated if the ability to detect whether a task has been accomplished is limited. For example, if task B highly depends on task A then until A is accomplished, all efforts invested in B may be completely wasted. How should one divide the effort between these tasks if feedback on the success of A is not available?

In this preliminary work we propose a setting that captures some of the fundamental challenges that are inherent to the process of decision making without feedback. We introduce the *dependent doors* problem, informally described as follows. There are $d \geq 2$ doors (representing tasks) which are initially closed, and the aim is to open all of them as fast as possible. To open a door, the algorithm can "knock" on it and it might open or not according to some governing probability distribution, that may depend on other doors being open or closed[1]. We focus on settings in which doors are positively correlated, which informally means that the probability of opening a door is never decreased if another door is open. The governing distributions and their dependencies are known to the algorithm in advance. Crucially, however, during the execution, it gets no direct feedback on whether or not a door has opened unless all d doors have opened, in which case the task is completed.

This research has actually originated from our research on heuristic search on trees [4]. Consider a tree of depth d with a treasure placed at one of its leaves. At each step the algorithm can "check" a vertex, which is child of an already checked vertex. Moreover, for each level of the tree, the algorithm has a way to compare the previously checked vertices on that level. This comparison has the property that if the ancestor of the treasure on that level was already checked, then it will necessarily be considered as the "best" on that level. Note, however, that unless we checked all the vertices on a given level, we can never be sure that the vertex considered as the best among checked vertices in the level is indeed the correct one. With such a guarantee, and assuming that the algorithm gets no other feedback from checked vertices, any reasonable algorithm that is about to check a vertex on a given level, will always choose to check a child of the current best vertex on the level above it. Therefore, the algorithm can be described as a sequence of levels to inspect. Moreover, if we know the different distributions involved, then we are exactly at the situation of the dependent doors problem. See the full version for more details on this example.

Another manifestation of d dependent doors can arise in the context of cryptography. Think about a sequence of d cascading encryptions, and separate decryption protocols to attack each of the encryptions. Investing more efforts in decrypting the i'th encryption would increase the chances of breaking it, but only if previous encryptions where already broken. On the other hand, we get no feedback on an encryption being broken unless all of them are.

The case of two doors can serve as an abstraction for exploration vs. exploitation problems, where it is typically the case that deficient performances on the exploration part may result in much waste on the exploitation part [10, 17]. It can also be seen as the question of balance between searching and verifying in algorithms that can be partitioned thus [1, 15]. In both

[1] Actually, the distribution associated with some door i may depend on the state of other doors (being open or closed) not only at the current knock, but also at the time of each of the previous knocks on door i.

examples, there may be partial or even no feedback in the sense that we don't know that the first procedure succeeded unless the second one also succeeds.

For simplicity, we concentrate on scenarios in which the dependencies are *acyclic*. That is, if we draw the directed dependency graph between doors, then this graph does not contain any directed cycles. The examples of searching and verifying and the heuristic search on trees can both be viewed as acyclic. Moreover, despite the fact that many configurations are not purely acyclic, one can sometimes obtain a useful approximation that is.

To illustrate the problem, consider the following presumably simple case of two dependent memoryless doors. The first door opens on each knock independently with probability 1/2. The second door can only open if the first door is open, in which case it opens on each knock independently, with probability 1/2. What is the sequence of knocks that minimizes the expected time to open both doors, remembering that we don't know when door 1 opens? It is easy to see that the alternating sequence $1, 2, 1, 2, 1, 2, \ldots$ results in 6 knocks in expectation. Computer simulations indicate that the best sequence gives a little more than 5.8 and starts with $1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2$. Applied to this particular scenario, our theoretical lower bound gives 5.747, and our upper bound gives a sequence with expected time 5.832.

1.1 Context and Related Work

This paper falls under the framework of decision making under uncertainty, a large research subject that has received significant amount of attention from researchers in various disciplines, including computer science, operational research, biology, sociology, economy, and even psychology and cognition, see, *e.g.,* [2, 3, 5, 6, 7, 8, 9, 16].

Performing despite limited feedback would fit the framework of reinforced learning [17] and is inherent to the study of exploration vs. exploitation type of problems, including Multi-Armed Bandit problems [10]. In this paper we study the impact of having no feedback whatsoever. Understanding this extreme scenario may serve as an approximation for cases where feedback is highly restricted, or limited in its impact. For example, if it turns out that the price of lacking feedback is small, then it may well be worth to avoid investing efforts in complex methods for utilizing the partial feedback.

Of particular interest is the case of two doors. As mentioned, difficulties resulting from the lack of feedback can arise when one aims to find a solution by alternating between two subroutines: Producing promising candidate solutions and verifying these candidates. Numerous strategies are based on this interplay, including heuristics based on brute force or trail and error approaches [1, 15], sample and predict approaches [11, 14, 17], iterative local algorithms [12, 13], and many others. Finding strategies for efficiently balancing these two tasks can be therefore applicable.

1.2 Setting

There are $d \geq 2$ doors and each door can be either open or closed. Doors start closed, and once a door opens it never closes. To open a door, an algorithm can knock on it and it might open or not according to some probability distribution. The goal is to minimize the expected number of knocks until all doors open. Crucially, the algorithm has no feedback on whether or not a door has opened, unless all doors have opened, in which case the task is completed.

The probability that a door opens may depend on the state of other doors (being open or closed) at the time of the current knock as well as on their state during each of the previous knocks on the door. For example, the probability that a certain knock at door i succeeds may depend on the number of previous knocks on door i, but counting only those that were

made while some other specific door j was open. The idea behind this definition is that the more time we invest in opening a door the more likely it is to open, and the quality of each knock depends on what is the state of the doors it depends on at the time of the knock.

Below we provide a semi-formal description of the setting. The level of detail is sufficient to understand the content of the main text, which is mainly concerned with independent and cascading configurations. The reader interested in a more formal description of the model is referred to the full version.

A specific setting of doors is called a *configuration* (normally denoted $\mathcal{C}$). This includes a description of all dependencies between doors and the resulting probability distributions. In this paper we assume that the dependency graph of the doors is acyclic, and so we may assume that a configuration describes an ordering of the doors, such that each door depends only on lower index doors. Furthermore, we assume that the correlation between doors is positive, *i.e.*, a door being open can only improve the chances of other doors to open.

Perhaps the simplest configuration is when all doors are *independent* of each other. In this case, door i can be associated with a function $p_i : \mathbb{N} \to [0, 1]$, where $p_i(n)$ is the probability that door i is not open after knocking on it n times. Another family of acyclic configurations are *cascading* configurations. Here, door i cannot open unless all doors of lower index are already open. In this case, the configuration can again be described by a set of functions $\{p_i\}_{i=1}^{d}$, where $p_i(n)$ describes the probability that door i is not open after knocking on it n times, where the count starts only after door $i-1$ is already open.

In general, given a configuration, each door i defines a non-decreasing function $p_i : \mathbb{N} \to [0, 1]$, called the *fundamental distribution* of the door, where $p_i(n)$ is the probability that the door is not open after knocking on it n times in the best of conditions, *i.e.*, assuming all doors of lower index are open. In the case of independent and cascading configurations, the fundamental distribution p_i coincides with the functions mentioned above. Two doors are *similar* if they have the same fundamental distribution. Two configurations are *similar* if for every i, door i of the first configuration is similar to door i of the second.

When designing an algorithm, we will assume that the configuration it is going to run in is known. As there is no feedback, a deterministic algorithm can be thought of as a possibly infinite sequence of door knocks. A randomized algorithm is therefore a distribution over sequences, and as all of them will have expected running time at least as large as that of an optimal sequence (if one exists), the expected running time of a randomized algorithm cannot be any better. Denote by $\mathbb{T}_{\mathcal{C}}(\pi)$, the expected time until all doors open when running sequence π in configuration $\mathcal{C}$. We define $\mathbb{T}_{\mathcal{C}} = \min_{\pi} \mathbb{T}_{\mathcal{C}}(\pi)$. As proved in the full version of the paper, there exists a sequence achieving this minimum. Therefore, by the aforementioned arguments, we can restrict our discussion to deterministic algorithms only.

If we had feedback we would knock on each door until it opens, and then continue to the next. Denoting by $E_i = \sum_{n=0}^{\infty} p_i(n)$ the expected time to open door i on its own, the expected running time then does not depend on the specific dependencies between doors at all, and is $\sum_i E_i$. Also, this value is clearly optimal. To evaluate the impact of lacking feedback for a configuration $\mathcal{C}$, we therefore define:

$$\texttt{Price}(\mathcal{C}) = \frac{\mathbb{T}_{\mathcal{C}}}{\sum_i E_i} .$$

Obviously $\texttt{Price}(\mathcal{C}) \geq 1$, and for example, if all doors start closed and open after just 1 knock, it is in fact equal to 1. In the full version of this paper we also show that $\texttt{Price}(\mathcal{C}) \leq d$.

1.3 Our Results

We have two main results. The first one, presented in Section 2, states that any two similar configurations have the same optimal running time up to a constant factor. We stress that this constant factor is universal in the sense that it does not depend on the specific distributions or on the number of doors d.

Furthermore, given a configuration, we identify an algorithm that is optimal for it up to a constant factor. We then show that for configurations where all doors are similar, there is a much simpler algorithm which is optimal up to a constant factor, and describe a formula that computes its approximate running time. We conclude Section 2 by analysing the price of lacking feedback for several configurations governed by standard fundamental distributions. In particular, we show that the price is logarithmic in d for memoryless doors, but can potentially grow to be linear in d for other distributions.

We then turn our attention to identify exact optimal sequences. Perhaps the simplest case is the case of two cascading memoryless doors. That is, the first door opens on each knock independently with probability p_1. The second door can only open if the first door is open, in which case it opens on each knock independently, with probability p_2. In Section 3 we present our second main result: Algorithms for these configurations that achieve the precise optimal running time up to an additive term of 1.

On the technical side, to establish such an extremely competitive algorithm, we first consider a semi-fractional variant of the problem and find a sequence that achieves the precise optimal bound. We then approximate this semi-fractional sequence to obtain an integer solution losing only an additive term of 1 in the running time. A nice anecdote is that in the case where $p_1 = p_2$ and are very small, the ratio of 2-knocks over 1-knocks in the sequence we get approaches the golden ratio. Also, in this case, the optimal running time approaches $3.58/p_1$ as p_1 goes to zero. It follows that in this case, the price of lacking feedback tends to $3.58/2$ and the price of dependencies, *i.e.*, the multiplicative gap between the cascading and independent settings, tends to $3.58/3$.

2 Near Optimal Algorithms

The following important lemma is proved in the full version using a coupling argument:

▶ **Lemma 1.** *Consider similar configurations $\mathcal{C}, \mathcal{X}$ and $\mathcal{I}$, where $\mathcal{X}$ is cascading and $\mathcal{I}$ is independent. For every sequence π, $\mathbb{T}_{\mathcal{I}}(\pi) \leq \mathbb{T}_{\mathcal{C}}(\pi) \leq \mathbb{T}_{\mathcal{X}}(\pi)$. This also implies that $\mathbb{T}_{\mathcal{I}} \leq \mathbb{T}_{\mathcal{C}} \leq \mathbb{T}_{\mathcal{X}}$.*

The next theorem presents a near optimal sequence of knocks for a given configuration. In fact, by Lemma 1, this sequence is near optimal for any similar configuration, and so we get that the optimal running time for any two similar configurations is the same up to a universal multiplicative factor.

▶ **Theorem 2.** *There is a polynomial algorithm[2], that given a configuration $\mathcal{C}$ generates a sequence π such that $\mathbb{T}_{\mathcal{C}}(\pi) = \Theta(\mathbb{T}_{\mathcal{I}})$. In fact, $\mathbb{T}_{\mathcal{C}}(\pi) \leq 2 + 4\mathbb{T}_{\mathcal{I}} \leq 2 + 4\mathbb{T}_{\mathcal{C}}$.*

Proof. Denote by $p_1, \ldots, p_d$ the fundamental distributions of the doors of $\mathcal{C}$. For a finite sequence of knocks α, denote by $\mathtt{SC}_{\mathcal{C}}(\alpha)$ the probability that after running α in configuration

[2] A polynomial algorithm in our setting generates the next knock in the sequence in polynomial time in the index of the knock and in d, assuming that reading any specific value of any of the fundamental distributions of a door takes constant time.

$\mathcal{C}$, some of the doors are still closed. Note that if α is *sorted*, that is, if all knocks on door 1 are done first, followed by the knocks on doors 2, etc., then $\mathtt{SC}_{\mathcal{X}}(\alpha) = \mathtt{SC}_{\mathcal{I}}(\alpha)$.

We start by showing that for any T, we can construct in polynomial time a finite sequence α_T of length T that maximizes the probability that all doors will open, *i.e.*, minimizes $\mathtt{SC}_{\mathcal{I}}(\alpha_T)$. As noted above, if we sort the sequence, this is equal to $\mathtt{SC}_{\mathcal{X}}(\alpha_T)$.

The algorithm follows a dynamic programming approach, and calculates a matrix A, where $A[i,t]$ holds the maximal probability that a sequence of length t has of opening all of the doors $1, 2, \ldots, i$. All the entries $A[0, \cdot]$ are just 1, and the key point is that for each i and t, knowing all of the entries in $A[i, \cdot]$, it is easy to calculate $A[i+1, t]$:

$$A[i+1,t] = \max_{k=0}^{t} A[i, t-k] \cdot (1 - p_{i+1}(k)) \ .$$

Calculating the whole table takes $O(dT^2)$ time, and $A[d,T]$ will give us the highest probability a sequence of length T can have of opening all doors. Keeping tabs on the choices the max in the formula makes, we can get an optimal sequence α_T, and can take it to be sorted.

Consider the sequence $\pi = \alpha_2 \cdot \alpha_4 \cdots \alpha_{2^n} \cdots$. The complexity of generating this sequence up to place T is $O(dT^2)$, and so this algorithm is polynomial. Our goal will be to compare $\mathbb{T}_{\mathcal{X}}(\pi)$ with $\mathbb{T}_{\mathcal{I}}(\pi^\star)$, where $\pi^\star$ is the optimal sequence for $\mathcal{I}$.

The following observation stems from the fact that for any natural valued random variable X, $\mathtt{E}[X] = \sum_{n=0}^{\infty} \mathtt{Pr}[X > n]$ and $\mathtt{Pr}[X > n]$ is a non-increasing function of n.

▶ **Observation 3.** *Let $\{a_n\}_{n=1}^{\infty}$ be a strictly increasing sequence of natural numbers, and X be some natural valued random variable. Then:*

$$\sum_{n=1}^{\infty} (a_{n+1} - a_n) \mathtt{Pr}[X > a_{n+1}] \leq \mathtt{E}[X] \leq a_1 + \sum_{n=1}^{\infty} (a_{n+1} - a_n) \mathtt{Pr}[X > a_n] \ .$$

For a sequence π, denote by $\pi[n]$ the prefix of π of length n. In this terminology, $\mathbb{T}_{\mathcal{C}}(\pi) = \sum_{n=0}^{\infty} \mathtt{SC}_{\mathcal{C}}(\pi[n])$. Setting $a_n = 2 + 4 + \ldots + 2^n$ in the right side of Observation 3, and letting X be the number of rounds until all doors open when using π, we get:

$$\begin{aligned}
\mathbb{T}_{\mathcal{X}}(\pi) &\leq 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \mathtt{SC}_{\mathcal{X}}(\pi[2 + \ldots + 2^n]) \leq 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \mathtt{SC}_{\mathcal{X}}(\alpha_{2^n}) \\
&= 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \mathtt{SC}_{\mathcal{I}}(\alpha_{2^n}) \leq 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \mathtt{SC}_{\mathcal{I}}(\pi^\star[2^n]) \leq 2 + 4\mathbb{T}_{\mathcal{I}}(\pi^\star)
\end{aligned}$$

The last step is using Observation 3 with $a_n = 2^{n-1}$. Theorem 2 concludes. ◀

2.1 Configurations where all Doors are Similar

In this section we focus on configurations where all doors have the same fundamental distribution $p(n)$. We provide simple algorithms that are optimal up to a universal constant, and establish the price of lacking feedback with respect to a few natural distributions. Corresponding proofs appear in the full version of the paper.

2.1.1 Simple Algorithms

Let us consider the following very simple algorithm $A_{\mathtt{simp}}$. It runs in phases, where in each phase it knocks on each door once, in order. As a sequence, we can write $A_{\mathtt{simp}} = (1, 2, \ldots, d)^{\infty}$. Let $X_1, \ldots, X_d$ be i.i.d. random variables taking positive integer values, satisfying $\mathtt{Pr}[X_i > n] = p(n)$. The following is straightforward:

▶ **Claim 4.** $\mathbb{T}_{\mathcal{I}}(A_{\texttt{simp}}) = \Theta\left(d \cdot \mathtt{E}\left[\max\left\{X_1, \ldots, X_d\right\}\right]\right)$

This one is less trivial:

▶ **Claim 5.** *If all doors are similar then* $\mathbb{T}_{\mathcal{I}}(A_{\texttt{simp}}) = \Theta(\mathbb{T}_{\mathcal{I}})$

The claim above states that $A_{\texttt{simp}}$ is optimal up to a multiplicative constant factor in the independent case, where all doors are similar. As a result, we can also show:

▶ **Claim 6.** *Denote by* α_n *the sequence* $1^{2^n}, \ldots, d^{2^n}$. *If all doors are similar then for any configuration* $\mathcal{C}$, $\mathbb{T}_{\mathcal{C}}\left(\alpha_0 \cdot \alpha_1 \cdot \alpha_2 \cdots\right) = \Theta(\mathbb{T}_{\mathcal{C}})$.

In plain words, the above claim states that the following algorithm is optimal up to a universal constant factor for any configuration where all doors are similar: Run in phases where phase n consists of knocking 2^n consecutive times on each door, in order.

2.1.2 On the Price of Lacking Feedback

By Claims 4 and 5, investigating the price of lacking feedback when all doors are similar boils down to understanding the expected maximum of i.i.d. random variables.

$$\texttt{Price} = \Theta\left(\frac{\mathtt{E}\left[\max\left\{X_1, \ldots, X_d\right\}\right]}{\mathtt{E}\left[X_1\right]}\right) \tag{1}$$

Note that we omitted dependency on the configuration, as by Theorem 2, up to constant factors, it is the same price as in the case where the doors are independent. Let us see a few examples of this value. First:

▶ **Lemma 7.** *If* $X_1, \ldots, X_d$ *are i.i.d. random variables taking natural number values, then:*

$$\mathtt{E}\left[\max(X_1, \ldots, X_d)\right] = \Theta\left(\kappa + d\sum_{n=\kappa}^{\infty} \mathtt{Pr}\left[X_i > n\right]\right)$$

Where $\kappa = \min\left\{n \in \mathbb{N} \,|\, \mathtt{Pr}\left[X_1 > n\right] < 1/d\right\}$

▶ **Example 8.** After the first knock on it, each door opens with probability $1 - 1/d$ and if it doesn't, it will open at its $d+1$'st knock. The expected time to open each door on its own is 2. By Lemma 7, as $\kappa = d + 1$, we get that $\texttt{Price} = \Omega(\kappa) = \Omega(d)$. Since always $\texttt{Price} \leq d$, $\texttt{Price} = \Theta(d)$.

▶ **Example 9.** If $p(n) = q^n$ for some $1/2 < q < 1$, then $\texttt{Price} = \Theta(\log(d))$.

▶ **Example 10.** If for some $c > 0$ and $a > 1$, $p(n) = \min(1, c/n^a)$, then $\texttt{Price} = \Theta(d^{\frac{1}{a}})$.

Sometimes we know a bound on some moment of the distribution of opening a door. If $\mathtt{E}\left[X_1\right] < M$, since $\texttt{Price} \leq d$, then $\mathbb{T} = O(d^2 M)$. Also,

▶ **Example 11.** If $\mathtt{E}\left[X_1^a\right] < M$ for some $a > 1$, then $\mathbb{T} = O\left(d^{1+\frac{1}{a}} M^{1/a}(1 + \frac{1}{a-1})\right)$.

For example, if the second moment of the time to open a door on its own is bounded, we get an $O(d^{3/2})$ algorithm.

3 Two Memoryless Cascading Doors

One can say that by Theorem 2 we solved much of the dependent doors problem. There is an equivalence of the independent and cascading models, and we give an up to constant factor optimal algorithm for any situation. However, we still find the question of finding the true optimal sequences for cascading doors to be an interesting one. What is the precise cost of having no feedback, in numbers? Even the simple case of two doors, each opening with probability 1/2 on each knock, turns out to be quite challenging and has a not so intuitive optimal sequence.

In this section, we focus on a very simple yet interesting case of the cascading door problem, and solve it almost exactly. We have two doors. Door 1 opens with probability p_1 each time we knock on it, and door 2 opens with probability p_2. We further extend the setting to consider different durations. Specifically, we assume that a knock on door 1 takes one time unit, and a knock on door 2 takes c time units. Denote $q_1 = 1 - p_1$ and $q_2 = 1 - p_2$. For brevity, we will call a knock on door 1 a *1-knock*, and a knock on door 2 a *2-knock*.

The Semi-Fractional Model. As finding the optimal sequence directly proved to be difficult, we introduce a relaxation of our original model, termed the *semi-fractional model*. In this model, we allow 1-knocks to be of any length. A knock of length t, where t is a non-negative real number, will have probability of $1 - q_1^t$ of opening the door. In this case, a sequence consists of the alternating elements 1^t and 2, where 1^t describes a knock of length t on door 1. We call sequences in the semi-fractional model *semi-fractional sequences*, and to differentiate, we call sequences in the original model *integer sequences*.

As our configuration $\mathcal{C}$ will be clear from context, for a sequence π, we define $\mathtt{E}[\pi] = \mathbb{T}_{\mathcal{C}}(\pi)$ to be the expected running time of the sequence. Clearly, every integer sequence has a similar semi-fractional sequence with the same expected running time. As we will see, the reverse is not far from being true. That being so, finding the optimal semi-fractional sequence will give an almost optimal integer sequence.

3.1 Equivalence of Models

▶ **Theorem 12.** *Every semi-fractional sequence π has an integer sequence π', s.t.,* $\mathtt{E}[\pi'] \leq \mathtt{E}[\pi] + 1$.

For this purpose, in this subsection only, we describe a semi-fractional sequence π as a sequence of non-decreasing non-negative real numbers: $\pi_0, \pi_1, \pi_2, \ldots$, where $\pi_0 = 0$. This sequence describes the following semi-fractional sequence (in our original terms):

$$1^{\pi_1 - \pi_0} \cdot 2 \cdot 1^{\pi_2 - \pi_1} \cdot 2 \cdots$$

This representation simplifies our proofs considerably. Here are some observations:

- 1-knocks can be of length 0, yet we still consider them in our indexing.
- The sequence is an integer sequence iff for all i, $\pi_i \in \mathbb{N}$.
- The i-th 2-knock starts at time $\pi_i + c(i-1)$ and ends at $\pi_i + ci$.
- The probability of door 1 being closed after the completion of the i-th 1-knock is $q_1^{\pi_i}$, and so the probability it opens at 1-knock i is $q_1^{\pi_{i-1}} - q_1^{\pi_i}$

▶ **Lemma 13.** *For two sequences $\pi = (\pi_0, \pi_1, \ldots)$ and $\pi' = (\pi'_1, \pi'_2, \ldots)$, if for all i, $\pi_i \leq \pi'_i \leq \pi_i + 1$ then $\mathtt{E}[\pi'] \leq \mathtt{E}[\pi] + 1$.*

Lemma 13 is the heart of our theorem. Indeed, once proven, Theorem 12 follows in a straightforward manner. Given a semi-fractional sequence π, define $\pi'_i = \lceil \pi_i \rceil$. Then, π' is an integer sequence, and it satisfies the conditions of the lemma, so we are done. The lemma makes sense, as the sequence π' in which for all $i > 0$, $\pi'_i = \pi_i + 1$, can be thought of as adding a 1-knock of length one in the beginning of the sequence. Even if this added 1-knock did nothing, the running time would increase by at most 1. However, the proof is more involved, since in the lemma, while some of the 2-knocks may have an increased chance of succeeding, some may actually have a lesser chance.

Proof. Given a sequence π and an event X, we denote by $\mathtt{E}[\pi \mid X]$ the expected running time of π given the event X. Let X_i denote the event that door 1 opens at its i-th 1-knock. As already said:

$$\mathtt{Pr}[X_i] = q_1^{\pi_{i-1}} - q_1^{\pi_i} = \int_{\pi_{i-1}}^{\pi_i} q_1^x \ln(q_1)\,\mathrm{d}x$$

Where the last equality comes as no surprise, as it can be seen as modelling door 1 in a continuous fashion, having an exponential distribution fitting its geometrical one. Now:

$$\mathtt{E}[\pi] = \sum_{i=1}^{\infty} \mathtt{Pr}[X_i]\,\mathtt{E}[\pi \mid X_i] = \sum_{i=1}^{\infty} \int_{\pi_{i-1}}^{\pi_i} q_1^x \ln(q_1)\,\mathrm{d}x \cdot \mathtt{E}[\pi \mid X_i] = \int_0^{\infty} q_1^x \ln(q_1) \cdot \mathtt{E}\left[\pi \,\middle|\, X_{i(x)}\right] \mathrm{d}x$$

Where $i(x) = \max_i \{x \geq \pi_{i-1}\}$, that is, the index of the 1-knock that x belongs to when considering only time spent knocking on door 1. Defining X'_i and $i'(x)$ in an analogous way for π', we want to show that for all x,

$$\mathtt{E}\left[\pi' \,\middle|\, X'_{i'(x)}\right] \leq 1 + \mathtt{E}\left[\pi \,\middle|\, X_{i(x)}\right]$$

as using it with the last equality will prove the lemma. We need the following three claims:

1. If $j \leq i$, then $\mathtt{E}[\pi \mid X_j] \leq \mathtt{E}[\pi \mid X_i]$
2. For all x, $i'(x) \leq i(x)$
3. For all i, $\mathtt{E}[\pi' \mid X'_i] \leq 1 + \mathtt{E}[\pi \mid X_i]$

Together they give what we need:

$$\mathtt{E}\left[\pi' \,\middle|\, X'_{i'(x)}\right] \leq 1 + \mathtt{E}\left[\pi \,\middle|\, X_{i'(x)}\right] \leq 1 + \mathtt{E}\left[\pi \,\middle|\, X_{i(x)}\right]$$

The first is actually true trivially for all sequences, as the sooner the first door opens, the better the expected time to finish. For the second, since for all i, $\pi'_i \geq \pi_i$, then $x \geq \pi'_i$ implies that $x \geq \pi_i$, and so:

$$i'(x) = \max_i \left\{x \geq \pi'_{i-1}\right\} \leq \max_i \{x \geq \pi_{i-1}\} = i(x)$$

For the third, denote by Y_j the event that door 2 opens at the j'th 2-knock. Then:

$$\mathtt{E}[\pi \mid X_i] = \sum_{j=i}^{\infty} (\pi_j + cj)\mathtt{Pr}[Y_j \mid X_i]$$

Let us consider this same expression as it occurs in π'. First note that $\mathtt{Pr}[Y_j \mid X_i] = \mathtt{Pr}\left[Y'_j \,\middle|\, X'_i\right]$, as all that matters for its evaluation is $j - i$. Therefore:

$$\begin{aligned}\mathtt{E}[\pi' \mid X'_i] &= \sum_{j=i}^{\infty} (\pi'_j + cj)\mathtt{Pr}\left[Y'_j \,\middle|\, X'_i\right] \leq \sum_{j=i}^{\infty} (\pi_j + 1 + cj)\mathtt{Pr}[Y_j \mid X_i] \\ &= \mathtt{E}[\pi \mid X_i] + \sum_{j=i}^{\infty} \mathtt{Pr}[Y_j \mid X_i] \leq \mathtt{E}[\pi \mid X_i] + 1\,.\end{aligned}$$

◀

3.2 The Optimal Semi-Fractional Sequence

A big advantage of the semi-fractional model is that we can find an optimal sequence for it. For that we need some preparation:

▶ **Definition 14.** For a semi-fractional sequence π, and some $0 \leq x \leq 1$, denote by $\mathtt{E}_x[\pi]$ the expected running time of π when started with door 1 being closed with probability x. In this notation, $\mathtt{E}[\pi] = \mathtt{E}_1[\pi]$.

▶ **Lemma 15.** *Let $y = x/(q_2 + p_2 x)$. Then:*

$$\mathtt{E}_x\left[1^t \cdot \pi\right] = t + \mathtt{E}_{q_1^t x}[\pi] \qquad \mathtt{E}_x[2 \cdot \pi] = c + \frac{x}{y}\mathtt{E}_y[\pi]$$

Proof. The first equation is clear, since starting with door 1 being closed with probability x, and then knocking on it for t rounds, the probability that this door is closed is $q_1^t x$.

As for the second equation, if door 1 is closed with probability x, then knocking on door 2, we have a probability of $p_2(1-x)$ of terminating, and so the probability we did not finish is:

$$1 - p_2(1-x) = 1 - p_2 + p_2 x = q_2 + p_2 x = \frac{x}{y}$$

It remains to show that conditioning on the fact that we indeed continue, the probability that door 1 is closed is y. It is the following expression, evaluated after a 2-knock:

$$\frac{\mathtt{Pr}[\text{door 1 is closed}]}{\mathtt{Pr}[\text{door 1 is closed}] + \mathtt{Pr}[\text{door 1 is open but not door 2}]} = \frac{x}{x + (1-x)q_2} = y\,. \qquad \blacktriangleleft$$

Applying Lemma 15 iteratively on a finite sequence w, we get:

$$\mathtt{E}_x[w\pi] = a(x,w) + b(x,w)\mathtt{E}_{\delta(x,w)}[\pi] \tag{2}$$

Of specific interest is $\delta(x,w)$. It can be thought of as the *state*[3] of our algorithm after running the sequence w, when we started at state x. Lemma 15 and Equation (2) give us the behaviour of $\delta(x,w)$:

$$\delta(x,1^t) = q_1^t x\,, \qquad \delta(x,2) = \frac{x}{q_2 + p_2 x}\,, \qquad \delta(x,aw) = \delta(\delta(x,a),w)\,.$$

We start with the state being 1, since we want to calculate $\mathtt{E}_1[\pi]$. Except for this first moment, as we can safely assume any reasonable algorithm will start with a 1-knock, the state will always be in the interval $(0,1)$. A 1-knock will always decrease the state and a 2-knock will increase it.

Our point in all this, is that we wish to exploit the fact that our doors are memoryless, and if we encounter a state we've already been at during the running of the sequence, then we should probably make the same choice now as we did then. The following definition and lemma capture this point.

▶ **Definition 16.** We say a non-empty finite sequence w is x-invariant, if $\delta(x,w) = x$.

The following Lemma is proved in the full version of this paper, and formalizes our intuition about how an optimal algorithm should behave.

▶ **Lemma 17.** *If w is x-invariant, and $\mathtt{E}_x[w\pi] \leq \mathtt{E}_x[\pi]$ then $\mathtt{E}_x[w^\infty] \leq \mathtt{E}_x[w\pi]$.*

[3] There is an intuitive meaning behind this. Going through Lemma 15, we can see that $\delta(1,w)$ is actually the probability that after running w, door 1 is closed conditioned on door 2 being closed. Indeed, After running some finite sequence, the only feedback we have is that the algorithm did not finish yet. We can therefore calculate from our previous moves what is the probability that door 1 is closed, and that is the only information we need for our next steps.

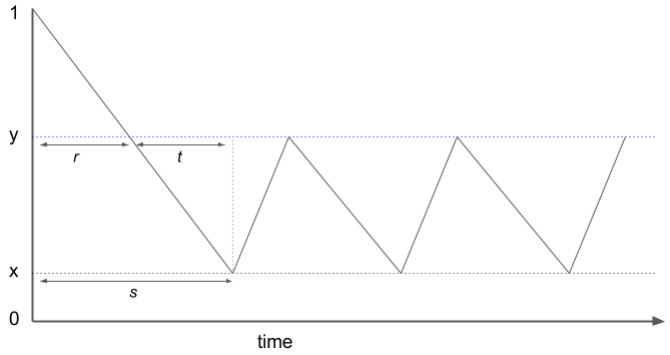


Figure 1 How the state evolves as a function of time. 1-knocks decrease the state, and 2-knocks increase it. Note that $r = \log_{q_1}(y)$ and $s = \log_{q_1}(x)$.

3.2.1 The Actual Semi-Fractional Sequence

▶ **Theorem 18.** *There is an optimal semi-fractional sequence $\pi^\star$ of the form $1^s(21^t)^\infty$, for some positive real values s and t, and its running time is:*

$$\mathtt{E}\left[\pi^\star\right] = \min_{z\in[0,1]}\left(\log_{q_1}(1-z) + \frac{c + (1-p_2 z)\log_{q_1}(1-p_2 z)}{p_2 z}\right).$$

Proof. In the full version of this paper, we prove that there is an optimal semi-fractional sequence π. It clearly starts with a non-zero 1-knock, and so we can write $\pi = 1^s 2\pi'$. Intuitively, in terms of its state, this sequence starts at 1, goes down for some time with a 1-knock, and then jumps back up with a 2-knock. The state it reaches now was already passed through on the first 1-knock, and so as this is an optimal sequence we can assume it will choose the same as it did before, and keep zig-zaging up and down.

We next prove that indeed there is an optimal sequence following the zig-zaging form above. Again, take some optimal π, and write $\pi = 1^s 2\pi'$. Denote $x = \delta(1, 1^s)$ and $y = \delta(1, 1^s 2) = \delta(x, 2) > x$ (see Figure 1). Taking $r = \log_{q_1}(y) < s$, we get $\delta(1, 1^r) = y$. Denoting $t = s - r$, this means that $1^t 2$ is y-invariant. Since π is optimal, then:

$$\mathtt{E}\left[\pi\right] = \mathtt{E}\left[1^r(1^t 2)\pi'\right] \leq \mathtt{E}\left[1^r\pi'\right] \quad \text{which implies:} \quad \mathtt{E}_y\left[1^t 2\pi'\right] \leq \mathtt{E}_y\left[\pi'\right].$$

So by Lemma 17:

$$\mathtt{E}_y\left[(1^t 2)^\infty\right] \leq \mathtt{E}_y\left[1^t 2\pi'\right] \quad \text{which implies:} \quad \mathtt{E}\left[1^r(1^t 2)^\infty\right] \leq \mathtt{E}\left[1^r 1^t 2\pi'\right] = \mathtt{E}\left[\pi\right].$$

Therefore, $1^r(1^t 2)^\infty = 1^s(21^t)^\infty$ is optimal. We denote this sequence $\pi^\star$.

Now for the analysis of the running time of this optimal sequence. We will use Lemma 15 many times in what follows.

$$\mathtt{E}_1\left[1^s(21^t)^\infty\right] = s + \mathtt{E}_x\left[(21^t)^\infty\right].$$

Denote $\alpha = (21^t)^\infty$.

$$\mathtt{E}_x\left[\alpha\right] = \mathtt{E}_x\left[21^t\alpha\right] = c + \frac{x}{y}\mathtt{E}_y\left[1^t\alpha\right] = c + \frac{x}{y}(t + \mathtt{E}_x\left[\alpha\right]).$$

Since $t = s - r = \log_{q_1}(x/y)$:

$$\mathtt{E}_x\left[\alpha\right] = \frac{c}{1-\frac{x}{y}} + \frac{\frac{x}{y}}{1-\frac{x}{y}}\log_{q_1}(x/y).$$

By Lemma 15, as our y is the state resulting from a 2-knock starting at state x, it follows that $y = x/(q_2 + p_2 x)$. Since $x/y = q_2 + p_2 x$, then $1 - x/y = p_2(1 - x)$ and then we get:

$$\frac{c}{p_2(1-x)} + \frac{q_2 + p_2 x}{p_2(1-x)} \log_{q_1}(q_2 + p_2 x) .$$

And in total:

$$\mathtt{E}_1\left[1^s(21^t)^\infty\right] = \log_{q_1}(x) + \frac{c + (q_2 + p_2 x)\log_{q_1}(q_2 + p_2 x)}{p_2(1-x)} .$$

Changing variable to $z = 1 - x$, results in $q_2 + p_2 x = 1 - p_2 z$, and we get the expression in the statement of the theorem. ◀

3.3 Actual Numbers

Theorem 18 gives the optimal semi-fractional sequence and a formula to calculate its expected running time. This formula can be approximated as accurately as we wish for any specific values of p_1, p_2 and c, but it is difficult to obtain a closed form formula from it. In the full version, we show an approximation with an additive error term $p2/\log(1/q_1)$. This is pretty close to p_2/p_1, and so when $p_1 \geq p_2$ it is just an additive error of 1.

In general, when p_1 is small, then the running time is shown to depend on $\theta \approx cp_1/p_2$, which is the expected time to open door 2 on its own, divided by the time to open door 1 on its own - a natural measure of the system. Then, ignoring the additive mistake, we show there that the lower bound is approximately $\mathcal{F}(\theta)/p_1$, where $\mathcal{F}$ is some function not depending on the parameters of the system. For example $\mathcal{F}(1) = 3.58$. So opening two similar doors without feedback when p is small takes about 3.58 times more time than opening one door as opposed to the case with feedback, where the factor is only 2.

We also note, that when the two doors are independent and similar, it is quite easy to see that the optimal expected running time is at most $3/p$. As a last interesting point, if $c = 1$ and $p = p_1 = p_2$ approaches zero, then the ratio between the number of 2-knocks and the number of 1-knocks approaches $\frac{1}{2}(1 + \sqrt{5})$, which is the golden ratio. These last two points are also shown in the full version of the paper.

3.4 Examples

For $p_1 = p_2 = 1/2$ and $c = 1$, the lower bound is 5.747. Simulations show that the best algorithm for this case is slightly more than 5.8, so the lower bound is quite tight, but our upper bound is 6.747 which is pretty far. However, the sequence we get from the upper bound proof starts with:

$$1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, \ldots$$

The value it gives is about 5.832, which is very close to optimal. For $p_1 = p_2 = 1/100$ and $c = 1$, the sequence we get is:

$$1^{97}, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, \ldots$$

And the value it gives is about 356.756, while the lower bound can be calculated to be approximately 356.754. As we see this is much tighter than the +1 that our upper bound promises.

References

1. Xiaohui Bei, Ning Chen, and Shengyu Zhang. On the complexity of trial and error. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 31–40, 2013. doi:10.1145/2488608.2488613.
2. David E. Bell. Regret in decision making under uncertainty. *Operations Research*, 30(5):961–981, 1982. doi:10.1287/opre.30.5.961.
3. Michael Ben-Or and Avinatan Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 221–230, 2008. doi:10.1109/FOCS.2008.58.
4. Lucas Boczkowski, Amos Korman, and Yoav Rodeh. Searching on trees with noisy memory. *CoRR*, abs/1611.01403, 2016. URL: http://arxiv.org/abs/1611.01403.
5. Matthias Brand, Christian Laier, Mirko Pawlikowski, and Hans J. Markowitsch. Decision making with and without feedback: The role of intelligence, strategies, executive functions, and cognitive styles. *Journal of Clinical and Experimental Neuropsychology*, 31(8):984–998, 2009. PMID: 19358007. doi:10.1080/13803390902776860.
6. Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 519–532, 2016. doi:10.1145/2897518.2897656.
7. Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, October 1994. doi:10.1137/S0097539791195877.
8. L. A. Giraldeau and T. Caraco. *Social Foraging Theory*. Monographs in behavior and ecology. Princeton University Press, 2000.
9. Richard M. Karp and Robert Kleinberg. Noisy binary search and its applications. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'07, pages 881–890, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1283383.1283478.
10. Michael N. Katehakis and Arthur F. Veinott, Jr. The multi-armed bandit problem: Decomposition and computation. *Math. Oper. Res.*, 12(2):262–268, May 1987. doi:10.1287/moor.12.2.262.
11. Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994.
12. Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. Toward more localized local algorithms: removing assumptions concerning global knowledge. *Distributed Computing*, 26(5-6):289–308, 2013. doi:10.1007/s00446-012-0174-8.
13. Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986. doi:10.1137/0215074.
14. Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.
15. Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2006.
16. Andrzej Pelc. Searching games with errors – fifty years of coping with liars. *Theor. Comput. Sci.*, 270(1-2):71–109, 2002. doi:10.1016/S0304-3975(01)00303-6.
17. Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

A Tight Lower Bound for the Capture Time of the Cops and Robbers Game*†

Sebastian Brandt[1], Yuval Emek[2], Jara Uitto[3], and Roger Wattenhofer[4]

1 **ETH Zürich, Zürich, Switzerland**
brandts@ethz.ch
2 **Technion, Haifa, Israel**
yemek@technion.ac.il
3 **ETH Zürich, Zürich, Switzerland; and University of Freiburg, Freiburg, Germany**
jara.uitto@inf.ethz.ch
4 **ETH Zürich, Zürich, Switzerland**
wattenhofer@ethz.ch

Abstract

For the game of *Cops and Robbers*, it is known that in 1-cop-win graphs, the cop can capture the robber in $O(n)$ time, and that there exist graphs in which this capture time is tight. When $k \geq 2$, a simple counting argument shows that in k-cop-win graphs, the capture time is at most $O(n^{k+1})$, however, no non-trivial lower bounds were previously known; indeed, in their 2011 book, Bonato and Nowakowski ask whether this upper bound can be improved. In this paper, the question of Bonato and Nowakowski is answered on the negative, proving that the $O(n^{k+1})$ bound is asymptotically tight for any constant $k \geq 2$. This yields a surprising gap in the capture time complexities between the 1-cop and the 2-cop cases.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases cops and robbers, capture time, lower bound

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.82

1 Introduction

The game of *Cops and Robbers* is a perfect information two-player zero-sum game played on an undirected n-vertex graph $G = (V, E)$, where the first player is identified with $k \geq 1$ *cops*, indexed by the integers $0, \ldots, k-1$, and the second player is identified with a single *robber*. In round 0, the cop player chooses the initial (not necessarily distinct) cop locations $c_0(0), \ldots, c_{k-1}(0) \in V$ and following that, the robber player chooses the initial robber location $r(0) \in V$. Then, in round $t = 1, 2, \ldots$, the cop player chooses the next (not necessarily distinct) cop locations $c_0(t), \ldots, c_{k-1}(t) \in V$ under the constraint that $c_i(t) \in N^+(c_i(t-1))$ for every $0 \leq i \leq k-1$, where $N^+(v)$ denotes the neighborhood of vertex v in G including v itself; following that, the robber player chooses the next robber location $r(t) \in N^+(r(t-1))$.

The goal of the cop player is to ensure that $r(t-1) \in \{c_0(t), \ldots, c_{k-1}(t)\}$ for some finite round t, referred to as *capturing* the robber. Conversely, the goal of the robber player is to

* A full version of this paper can be found at http://www.disco.ethz.ch/publications/icalp2017-copsandrobbers.pdf.

† This work was partially supported by ERC Grant No. 336495 (ACDC).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 82; pp. 82:1–82:13

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

avoid being captured indefinitely. Graph G is said to be a *k-cop-win* graph if it admits a cop strategy $\mathcal{S}$ that guarantees capture. The *capture time* of $\mathcal{S}$ is defined to be the maximum number of rounds until capture is achieved, assuming optimal play by the robber. The *capture time* of graph G is then defined to be the minimum capture time of any cop strategy over G (notice that in this definition, it is assumed that k is clear from the context).

Bonato et al. [7] studied the capture time in single cop games and proved that every 1-cop-win graph admits a cop strategy that captures the robber in $\mathcal{O}(n)$ rounds. By considering a path, it is straightforward to verify that this bound is asymptotically tight. A simple configuration-counting argument (see, e.g., [5, 7]) shows that for any constant $k \geq 2$, if G is a k-cop-win graph, then its capture time is $\mathcal{O}(n^{k+1})$. One may suspect that this simple upper bound can be improved as it does not generalize the tight $\mathcal{O}(n)$ bound in the 1-cop setting. Answering an open question from Bonato and Nowakowski's book [9, Chapter 8], the main result of our paper is that perhaps surprisingly, this is not the case.

▶ **Theorem 1.** *There exist a universal positive constant α such that for every $k \geq 2$, there exists an infinite family $\mathcal{G}$ of k-cop-win graphs such that the capture time of any n-vertex graph $G \in \mathcal{G}$ is at least $(n/(\alpha k))^{k+1}$. Moreover, the smallest graph in $\mathcal{G}$ has $n = \mathcal{O}(k^2)$ vertices.*

Notice that for constant $k \geq 2$, this theorem provides an (existential) $\Omega(n^{k+1})$ lower bound on the capture time in k-cop-win graphs. Furthermore, it can be extended to non-constant values of $k = k(n)$ up to the conjectured maximum of $k(n) = \Theta(\sqrt{n})$ (see the related literature discussion), stating that in some k-cop-win graphs, the capture time is exponential in k and stretched exponential in n.

Related Literature

The Cops and Robbers game with a single cop was introduced by Quilliot [21] and independently by Nowakowski and Winkler [19] who also provided a full characterization of 1-cop-win graphs. This was generalized to the multiple cop setting by Aigner and Fromme [2] who defined the *cop number* of graph G to be the minimum number of cops that guarantees that the robber can be captured (that is, the minimum k for which G is a k-cop-win graph). Cast in this terminology, Aigner and Fromme proved that the cop number of any planar graph is at most 3. An upper bound of $\mathcal{O}(r^2)$ on the cop number of graphs excluding K_r as minor was established by Andreae [4]; this result lies at the heart of the recent graph decomposition technique of Abraham et al. [1] for the same family of graphs. For general graphs, the maximum possible cop number is still an open question: the famous Meyniel's Conjecture [14, 6] states that this number is $\Theta(\sqrt{n})$, where the state of the art is that it is bounded between $\Omega(\sqrt{n})$ [20] and $\mathcal{O}(n/2^{(1-o(1))\sqrt{\log n}})$ [17]. Several characterizations of graphs with cop number k are presented in [11].

As mentioned earlier, Bonato et al. [7] established a tight linear bound on the capture time in 1-cop-win graphs. For $k > 1$ cops, non-trivial bounds on the capture time in k-cop-win graphs were obtained mainly in the context of special graph classes, e.g., hypercubes [8] and Cartesian products of trees [18]. To the best of our knowledge, the linear lower bound of [7] is the (asymptotically) best previously known lower bound on the capture time in any class of graphs for any $k \geq 1$.

The capture time has been studied also for variants of the classic Cops and Robbers game. For example, the multiple robber setting was investigated by Förster et al. [13] who showed that the capture time may increase linearly with the number of robbers. Kehagias and Pralat [16] analyzed the expected capture time of a *drunk robber* whose strategy is

simply a random walk on the graph. For a broader overview of the results on the game of Cops and Robbers, the reader is referred to the book of Bonato and Nowakowski [9] and recent surveys [3, 10, 12, 15].

Techniques

Our lower bound proof relies on designing a bad (from the perspective of the cops) graph G that consists of several components, of which each has a different role (see the overview in Section 2.1). Here, we provide a glimpse into this design from an alternative (strictly informal and somewhat inaccurate) angle that may shed additional light on the techniques we use. At the heart of our construction lies the concept of forcing each entity (cop or robber) to follow a designated (non-simple) path in G, where in the scope of this discussion, we assume that every vertex in G admits a self-loop so that these paths may include null moves. Specifically, graph G contains equally long paths $\chi_0, \ldots, \chi_{k-1}$ and ρ, referred to in this discussion as the *canonical paths* of the cops and robber, respectively. The best strategy of the cop player is then to assign one cop, say Cop i, to each path χ_i so that $c_i(t) = \chi_i(t)$ for every t; in response to that, the best strategy of the robber is to play $r(t) = \rho(t)$. This induces a sequence σ of (distinct) configurations and the analysis is completed by showing that σ is sufficiently long and that the robber is captured only at its end.

The most challenging part in the design of such canonical paths is to prove that the aforementioned strategies are indeed optimal. To that end, we show that if the robber deviates from her canonical path at time t, then she is either captured immediately or the game shifts forward to a more advanced configuration $\sigma(t')$ for some $t' > t$. Conversely, if some cop deviates from her canonical path at time t, then the game shifts backwards to a less advanced configuration $\sigma(t')$ for some $t' < t$. The main feature in the latter argument is an *exit component* $\mathcal{X}$; if the robber manages to reach $\mathcal{X}$, then she can force the game to shift backwards to the beginning, i.e., to $\sigma(0)$. This threat is the key ingredient in the analysis of the cop strategy: we construct the cops' components so that they must strictly follow their canonical paths in order to cover all exits in $\mathcal{X}$.

Due to the verbosity of our construction and to the space limitations, we defer all the proofs and the discussion of the case of more than 2 cops to the full version.

Technical Terminology

We call the set $\{c_0(t), c_1(t), \ldots, c_{k-1}(t)\}$, for some t, a *cop combination*. We say that Cop i *covers* node v at time t if $v \in N^+(c_i(t))$. We may omit t if it is clear from the context. We extend this covering notion to more than one cop and more than one node, e.g., we say that *the* cops cover a set of nodes $\{v_1, v_2, \ldots, v_j\}$ if each of the v_i is covered by at least one of the cops. Also, we say that a node is covered, resp. *uncovered*, if at least one cop covers it, resp. if no cop covers it.

2 The Case of 2 Cops

2.1 Overview

In this section, we construct a family $\{G_{\hat{n}}\}_{\hat{n} \geq 3, \hat{n} \equiv 0 \pmod 3}$ of 2-cop-win graphs, where $\hat{n} \in \Theta(n)$.[1] Then, we show that the capture time for 2 cops in $G_{\hat{n}}$ is $\Omega(n^3)$ by giving an

[1] Throughout the paper, we denote the number of nodes of a graph by n.

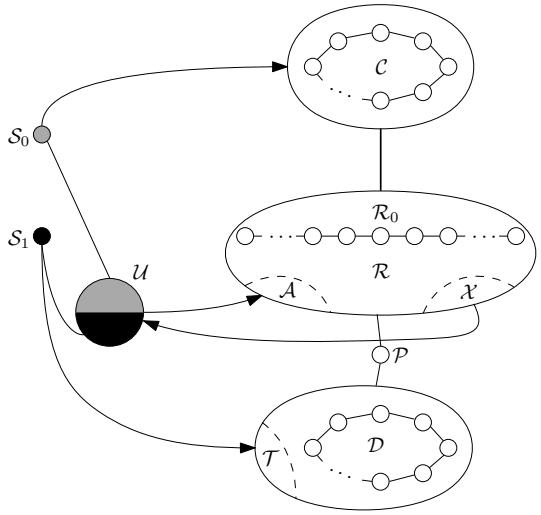


Figure 1 The graph $G_{\hat{n}}$ with its different components.

explicit strategy for the robber that achieves this capture time against any cop strategy. We conclude by presenting a cop strategy for 2 cops that achieves a capture time of $\mathcal{O}(n^3)$ against any robber strategy, which also serves as a proof that the graphs are indeed 2-cop-win. The other reason for explicitly specifying such a cop strategy (which *must* exist in 2-cop-win graphs, by the aforementioned simple configuration-counting argument) is that it forms the basis for a generalized cop strategy in the case of $k \geq 2$ cops. For a simplified overview of our graph construction, please refer to Figure 1.

The general idea behind the graph construction and the specified strategies for the cops and the robber is as follows: Our graph $G_{\hat{n}}$ contains a $\mathcal{U}$ component, where the robber cannot be captured, simply because each node in $\mathcal{U}$ has enough neighbors so that 2 cops cannot cover all of these neighbors simultaneously. Moreover, the robber can always stay in $\mathcal{U}$ (because each node in $\mathcal{U}$ has enough neighbors *in* $\mathcal{U}$) except if the cops go to two special nodes $\mathcal{S}_0$ and $\mathcal{S}_1$ which together cover all of $\mathcal{U}$.

When the robber is thereby flushed out of $\mathcal{U}$, she has to go to the $\mathcal{R}$ component of the graph. Note that the nodes of $\mathcal{R}$ induce a simple path on $G_{\hat{n}}$ and that after being flushed out of $\mathcal{U}$, the robber is located in the middle of this path. Now, the cops will continuously prevent the robber from escaping $\mathcal{R}$ and slowly force the robber to one end of the path where they will finally capture her. In order for this to take a long time, each node in $\mathcal{R}$ is connected to a set of so-called *exits* which are nodes that together form the $\mathcal{X}$ component of the graph. If the robber should manage to get to some exit, then she will be able to go back to her preferred $\mathcal{U}$ component, unless the cops go again to the special $\mathcal{S}_0$ and $\mathcal{S}_1$ nodes, in which case the robber can go back to the middle of the $\mathcal{R}$ path and thereby revert to a previous configuration. Hence, in order to capture the robber, the cops have to continuously cover these exits.

Unfortunately for the cops, there are only a few cop combinations that actually cover all exits of a node in $\mathcal{R}$. Moreover, only some of these cop combinations are *proper* in the sense that they also prevent the robber from moving back on the $\mathcal{R}$ path towards the middle which is essential for the cops in order to capture the robber. These proper *exit-covering* cop combinations are described in the following. For an illustration of the underlying graph structure, we refer to Figure 2.

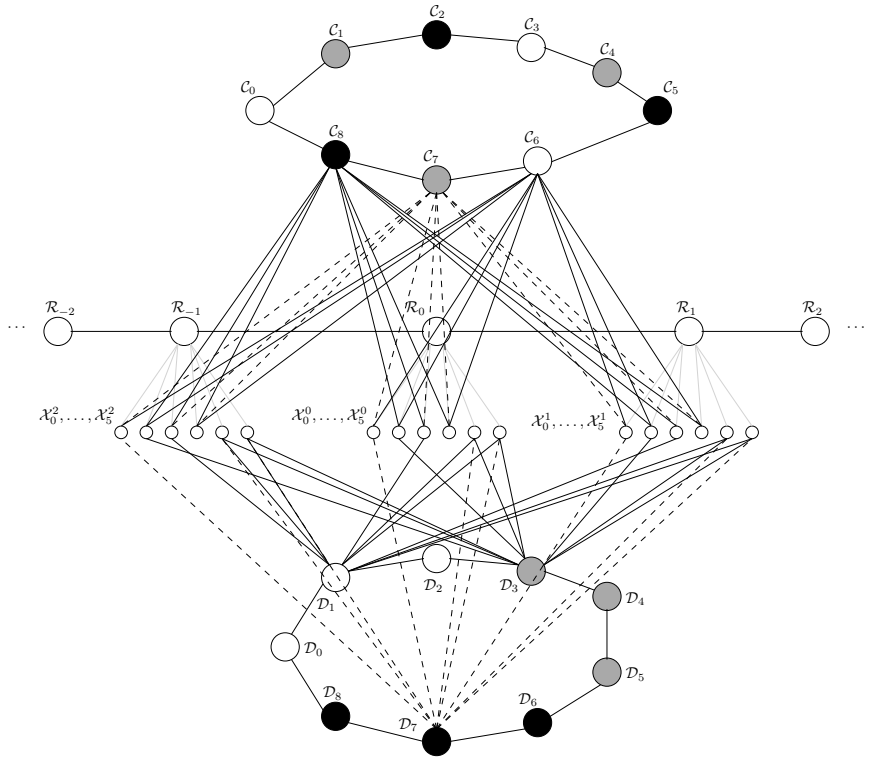


Figure 2 A part of the structure of $G_{\hat{n}}$ around the $\mathcal{X}$ nodes. A $\mathcal{C}$ node and a $\mathcal{D}$ node together cover the exits of an $\mathcal{R}$ node if and only if they have the same color.

One cop, say Cop 0, has to be in the $\mathcal{C}$ component of $G_{\hat{n}}$ and the other one (Cop 1) in the $\mathcal{D}$ component. The nodes in the $\mathcal{C}$ (resp., $\mathcal{D}$) component induce a simple cycle on $G_{\hat{n}}$. Assume for simplicity that the number of nodes in these two cycles are both multiples of 3. Now, we can imagine that the nodes in the $\mathcal{C}$ cycle are consecutively colored $0, 1, 2, 0, 1, 2, \ldots$ and that the nodes in the $\mathcal{D}$ cycle are colored $0, \ldots, 0, 1, \ldots, 1, 2, \ldots, 2$, resulting in three equally-sized monochromatic blocks. Now, the nodes in $\mathcal{C}$ and $\mathcal{D}$ are connected to the nodes in $\mathcal{X}$ in so that Cop 0 (in $\mathcal{C}$) and Cop 1 (in $\mathcal{D}$) cover all exits of an $\mathcal{R}$ node if and only if the nodes the two cops are occupying have the same color. Thus, if Cop 0 wants to move, e.g., clockwise, in her $\mathcal{C}$ cycle, then between any two consecutive steps, she has to wait for Cop 1 to travel roughly a third of her $\mathcal{D}$ cycle.

Similarly, using an independent color pallet, we color the nodes along the $\mathcal{R}$ path $3, 4, 5, 3, 4, 5, \ldots$ and color the nodes along the $\mathcal{C}$ cycle $3, \ldots, 3, 4, \ldots, 4, 5, \ldots, 5$. To prevent the robber from moving back towards the middle of her $\mathcal{R}$ path, we construct $G_{\hat{n}}$ so that a $\mathcal{C}$ node covers an $\mathcal{R}$ node if and only if they do *not* have the same color. Thus, if Cop 0 proceeds along her $\mathcal{C}$ cycle, then as soon as the color of the $\mathcal{C}$ node changes, the robber is forced to move one step forward along the $\mathcal{R}$ path. This accounts for Cop 0 traversing roughly a third of the $\mathcal{C}$ cycle for each step of the robber along the $\mathcal{R}$ path. The direction of the robber's movement (towards either end of the $\mathcal{R}$ path) is determined by the direction (clockwise or counterclockwise) of the movement of Cop 0 along the $\mathcal{C}$ cycle. We refer to Figure 3 for an illustration of how the robber is pushed along the $\mathcal{R}$ by a cop residing in $\mathcal{C}$.

Now, we design the graph $G_{\hat{n}}$ so that the $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{R}$ components consist of roughly $\hat{n}$ nodes. Thus, the robber takes $\Omega(\hat{n})$ steps until she is captured, for each of her steps Cop 0

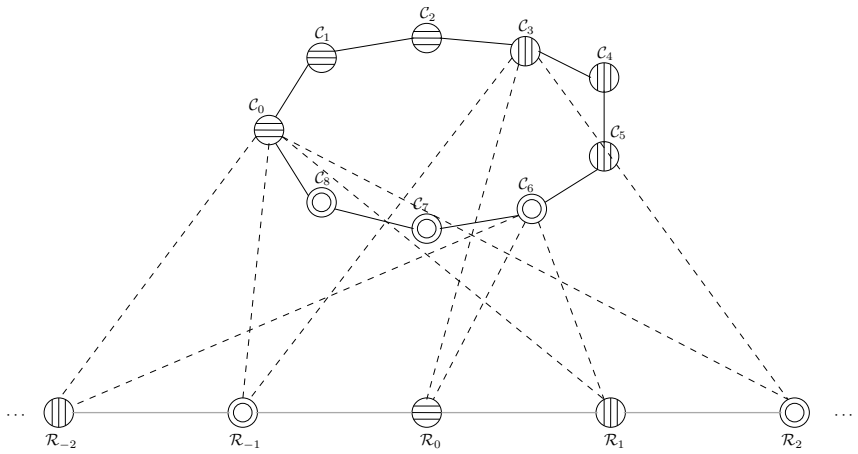


Figure 3 The edge structure between the $\mathcal{C}$ nodes and the $\mathcal{R}$ nodes. If a cop is in a $\mathcal{C}$ node and the robber in an $\mathcal{R}$ node, then the robber has to make sure that that these two nodes have the same color in order to avoid capture in the next move. By circling clockwise in $\mathcal{C}$, the cop can force the robber towards one end of the $\mathcal{R}$ path, by circling counterclockwise towards the other.

has to take $\Omega(\hat{n})$ steps, and for each step of Cop 0, Cop 1 has to take $\Omega(\hat{n})$ steps, resulting in a total capture time of $\Omega(\hat{n}^3)$. Since every component of the $G_{\hat{n}}$ except $\mathcal{R}$, $\mathcal{C}$ and $\mathcal{D}$ is of constant size, $\hat{n}$ is linear in the number n of nodes. Hence, we obtain a capture time of $\Omega(n^3)$ for the graph class $\{G_{\hat{n}}\}_{\hat{n}\geq 3,\hat{n}\equiv 0 \pmod 3}$.

Before proceeding to the exact details of the graph construction, four remarks are in order. Firstly, when the robber is flushed out of $\mathcal{U}$ to $\mathcal{R}$, there is an intermediate step between leaving $\mathcal{U}$ and arriving in $\mathcal{R}$ for technical reasons: Flushed out of $\mathcal{U}$, the robber actually has to go to a graph component called $\mathcal{A}$. Then, one cop moves to another graph component denoted $\mathcal{T}$ from which she covers all of $\mathcal{A}$ while ensuring that all nodes in $\mathcal{U}$ the robber could go to are covered. Thus, the robber is flushed out of $\mathcal{A}$ and is forced to go to $\mathcal{R}$ so that we can proceed as explained above.

Secondly, since Cop 0 has to be able to cover nodes from $\mathcal{R}$ when she is in $\mathcal{C}$, there have to be edges between $\mathcal{R}$ and $\mathcal{C}$. To prevent the robber from escaping $\mathcal{R}$ by going to a node in $\mathcal{C}$ via one of these edges, we augment the graph with a special $\mathcal{P}$ node. If, after the robber indeed moves to the $\mathcal{C}$ component, Cop 1 moves to $\mathcal{P}$ and Cop 0 moves to $\mathcal{S}_0$, then together they cover all $\mathcal{C}$ nodes and all their neighbors, ensuring that the robber will be captured in the next round.

Thirdly, the robber does not have to start in $\mathcal{U}$, but in fact it is best for her if she does (in the sense of increasing the capture time against the best cop strategy), provided that she cannot be captured immediately. In turn, this means that the cops should start in $\mathcal{S}_0$ and $\mathcal{S}_1$ in order to force the robber to start elsewhere. Even if the robber starts elsewhere, the cop strategy explained above forces the robber to the $\mathcal{R}$ component. Moreover, the robber can simply start in the middle of the $\mathcal{R}$ path and the cops cannot avoid having to go through the exit-covering routine explained above.

Fourthly, while the cop strategy explained above chases the robber from the middle of the $\mathcal{R}$ path to one of its ends, for simplicity, we will formally present a slightly simplified version where the end of the path the robber is chased to is fixed in advance (so the robber will be chased from one end of the path to the other end in the worst case for the cops).

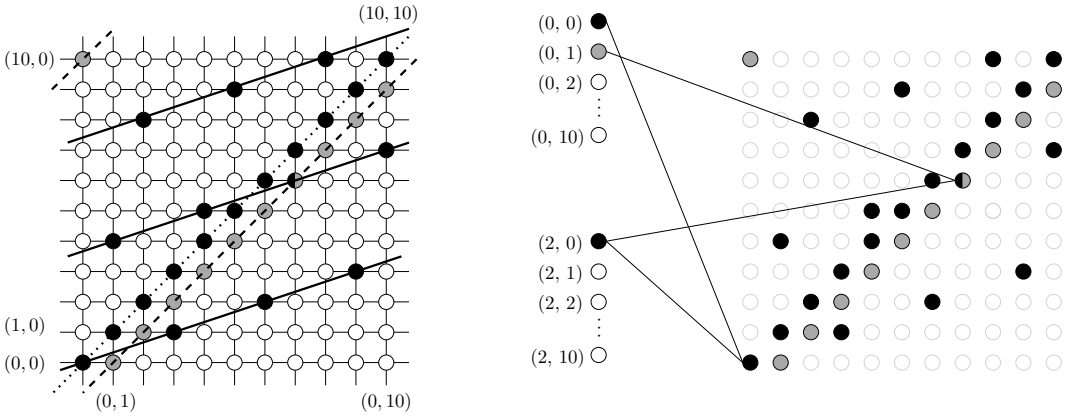


Figure 4 The construction of $\mathcal{U}$. On the left side, we see the 11×11 grid constituting $\mathcal{E}$ and three example lines $\mathcal{L}_{0,0}$, $\mathcal{L}_{0,1}$ and $\mathcal{L}_{2,0}$ from $\mathcal{L}$. On the right side, we see the nodes of the bipartite graph $G_{\mathcal{E},\mathcal{L}}$ where the left-hand column of nodes constitutes (a part of) one side of the bipartition, $\mathcal{L}$, and the right-hand grid the other side, $\mathcal{E}$. The edges between $\mathcal{E}$ and $\mathcal{L}$ are determined by the incidence relation of the nodes and lines in the left-hand 11×11 grid.

2.2 The Graph Construction

As explained above, the graphs $G_{\hat{n}}$ we are about to construct contain a component $\mathcal{U}$ in which the robber cannot be captured and from which the robber can only be flushed out by a specific cop combination outside of $\mathcal{U}$. Hence, the subgraph of any $G_{\hat{n}}$ induced by (the respective) $\mathcal{U}$ cannot be a 2-cop-win graph. As we want to generalize our graph construction to the case of more than 2 cops, we thus need a way to construct a graph where k cops cannot capture the robber. For this, inspired by the use of projective planes for constructing graphs with high cop numbers in [20], we will use incidence graphs of objects resembling affine planes. An explicit construction (for the case of 2 cops) is given in the following. For an illustration of the construction, we refer to Figure 4.

Let $\mathcal{E} = \{\mathcal{E}_{i,j} \mid 0 \leq i \leq 10 \wedge 0 \leq j \leq 10\}$ be a set of *elements* which we can imagine as arrayed in an 11×11 grid. Let $\mathcal{L} = \{\mathcal{L}_{i,j} \mid 0 \leq i \leq 9 \wedge 0 \leq j \leq 10\}$ be a set of *lines* where each $\mathcal{L}_{i,j}$ is defined as $\mathcal{L}_{i,j} = \{\mathcal{E}_{h,h(i+1)+j \pmod{11}} \mid 0 \leq h \leq 10\}$. Thus, each line $\mathcal{L}_{i,j}$ may be considered as a "line modulo 11" in our grid which goes through the element $\mathcal{E}_{0,j}$ and whose slope is determined by the parameter i (or more precisely $i+1$).

Now consider the *incidence graph* $G_{\mathcal{E},\mathcal{L}}$ for $\mathcal{E}$ and $\mathcal{L}$ which is defined as follows: The nodes of $G_{\mathcal{E},\mathcal{L}}$ are exactly the elements and lines defined above, i.e., $V(G_{\mathcal{E},\mathcal{L}}) = \mathcal{E} \cup \mathcal{L}$, and there is an edge between some node $\mathcal{E}_{i,j}$ and some node $\mathcal{L}_{i',j'}$ if and only if $\mathcal{E}_{i,j}$ is contained in the set $\mathcal{L}_{i',j'}$ (i.e., if and only if $\mathcal{E}_{i,j}$ lies on the line $\mathcal{L}_{i',j'}$). There are no other edges, hence $G_{\mathcal{E},\mathcal{L}}$ is bipartite where one side of the bipartition is given by $\mathcal{E}$ and the other side by $\mathcal{L}$.

▶ **Lemma 2.** *In $G_{\mathcal{E},\mathcal{L}}$, any two nodes in $\mathcal{E}$ have at most one common neighbor in $\mathcal{L}$. Also, any two nodes in $\mathcal{L}$ have at most one common neighbor in $\mathcal{E}$.*

▶ **Lemma 3.** *Let $i \in \{0, \ldots, 10\}$ be fixed. Any node from $\mathcal{L}$ has exactly one neighbor in $G_{\mathcal{E},\mathcal{L}}$ of the form $\mathcal{E}_{i,j}$ and exactly one neighbor in $G_{\mathcal{E},\mathcal{L}}$ of the form $\mathcal{E}_{j,i}$.*

▶ **Lemma 4.** *Let $i \in \{0, \ldots, 9\}$ be fixed. Any node from $\mathcal{E}$ has exactly one neighbor in $G_{\mathcal{E},\mathcal{L}}$ of the form $\mathcal{L}_{i,j}$.*

For the construction of $G_{\hat{n}}$, we will *borrow* nodes from $G_{\mathcal{E},\mathcal{L}}$ and we will assume that the borrowed nodes take along their relationship concerning edges between them, i.e., there is an

edge between two borrowed nodes in our graph construction if and only if there is an edge between those two nodes in $G_{\mathcal{E},\mathcal{L}}$.

We construct $G_{\hat{n}}$ as given in the following. The vertex set of $G_{\hat{n}}$ is defined as the (disjoint) union of smaller vertex sets that constitute different parts of the graph with different purposes:

$$
\begin{aligned}
V(G_{\hat{n}}) &= \mathcal{E} \cup \mathcal{L}^* \cup \mathcal{S} \cup \mathcal{A} \cup \mathcal{T} \cup \mathcal{R} \cup \mathcal{C} \cup \mathcal{D} \cup \{\mathcal{P}\} \cup \mathcal{X}, \text{ where} \\
\mathcal{L}^* &= \{\mathcal{L}_{i,j} \mid 0 \leq i \leq 3 \wedge 0 \leq j \leq 10\} \\
\mathcal{S} &= \{\mathcal{S}_0, \mathcal{S}_1\} \\
\mathcal{A} &= \{\mathcal{A}_0, \ldots, \mathcal{A}_3\} \\
\mathcal{T} &= \{\mathcal{T}_0, \ldots, \mathcal{T}_3\} \\
\mathcal{R} &= \{\mathcal{R}_{-\hat{n}}, \ldots, \mathcal{R}_{\hat{n}}\} \\
\mathcal{C} &= \{\mathcal{C}_0, \ldots, \mathcal{C}_{\hat{n}-1}\} \\
\mathcal{D} &= \{\mathcal{D}_0, \ldots, \mathcal{D}_{\hat{n}-1}\} \\
\mathcal{X} &= \mathcal{X}^0 \cup \mathcal{X}^1 \cup \mathcal{X}^2 \\
\mathcal{X}^j &= \{\mathcal{X}^j_0, \ldots, \mathcal{X}^j_5\} \text{ for all } 0 \leq j \leq 2
\end{aligned}
$$

The edges of $G_{\hat{n}}$ are specified in Table 1. Moreover, we set $\mathcal{U} = \mathcal{E} \cup \mathcal{L}^*$. Furthermore, we call the nodes in $\mathcal{X}$ *exits*, and for each node $\mathcal{R}_i \in \mathcal{R}$ we call the nodes from $\mathcal{X}$, that are connected to $\mathcal{R}_i$, the *exits of* $\mathcal{R}_i$.

The node subsets $\mathcal{E}$ and $\mathcal{L}^*$ are borrowed from $G_{\mathcal{E},\mathcal{L}}$, but also the (renamed) nodes in $\mathcal{X}$ are borrowed from $G_{\mathcal{E},\mathcal{L}}$: We consider $\mathcal{X}$ as a subset of $\mathcal{L} \setminus \mathcal{L}^*$. To ensure that no node in $\mathcal{E}$ covers too many exits of some $\mathcal{R}$ node, the exits of any $\mathcal{R}$ node are borrowed (disjointly) from a set of $\mathcal{L} \setminus \mathcal{L}^*$ nodes of the same slope. More precisely,

$$
\begin{aligned}
&\mathcal{X}^0 \text{ is borrowed from } \{\mathcal{L}_{4,j} \mid 0 \leq j \leq 10\} \text{ ,} \\
&\mathcal{X}^1 \text{ is borrowed from } \{\mathcal{L}_{5,j} \mid 0 \leq j \leq 10\} \text{ ,} \\
&\mathcal{X}^2 \text{ is borrowed from } \{\mathcal{L}_{6,j} \mid 0 \leq j \leq 10\} \text{ .}
\end{aligned}
$$

As long as the above conditions are met, we do not care about the explicit choice of $\mathcal{X}$ as a subset of $\mathcal{L} \setminus \mathcal{L}^*$. We obtain the following corollary from Lemma 2:

▶ **Corollary 5.** *Any two nodes in $\mathcal{E}$ have at most one common neighbor in $\mathcal{L}^*$. Any two nodes in $\mathcal{L}^* \cup \mathcal{X}$ have at most one common neighbor in $\mathcal{E}$.*

2.3 Observations

Before specifying asymptotically best strategies for the robber and the cops in $G_{\hat{n}}$, we gather some useful observations about the structure of $G_{\hat{n}}$. In particular, we examine which cop combinations cover certain neighbors of certain nodes. We start by showing that the cops have to be in $\mathcal{S}_0$ and $\mathcal{S}_1$ in order to flush the robber out of $\mathcal{U}$.

▶ **Lemma 6.** *Consider any $u \in \mathcal{U}$. The only cop combination not containing u that covers all neighbors of u in $\mathcal{U}$ is $\{\mathcal{S}_0, \mathcal{S}_1\}$.*

We proceed by showing that if the robber has been flushed out of $\mathcal{U}$ to some node $\mathcal{A}_i$, then the cops can only make progress by going to $\{\mathcal{S}_0, \mathcal{T}_i\}$ because otherwise the robber can go back to $\mathcal{U}$ (or there is no progress if the cops simply stay in $\{\mathcal{S}_0, \mathcal{S}_1\}$).

▶ **Lemma 7.** *Consider any $\mathcal{A}_i$. The only cop combinations not containing $\mathcal{A}_i$ that cover all neighbors of $\mathcal{A}_i$ in $\mathcal{U}$ are $\{\mathcal{S}_0, \mathcal{S}_1\}$ and $\{\mathcal{S}_0, \mathcal{T}_i\}$.*

Table 1 A listing of the edges of $G_{\hat{n}}$. In each block, the nodes listed in the right column are the neighboring nodes of the node listed in the left column. For simplicity, we omit a separate block for specifying the neighbors of the $\mathcal{X}$ nodes. They can be inferred from the other blocks. The $\mathcal{X}$ nodes do not have any edges connecting them to each other since they are borrowed from the $\mathcal{L}$ part of $G_{\mathcal{E},\mathcal{L}}$.

Node	Neighbors
$\mathcal{E}_{i,j}$	$\mathcal{L}^*$ nodes as determined by $G_{\mathcal{E},\mathcal{L}}$ $\mathcal{S}_i \pmod 2$ $\mathcal{A}_j \pmod 4$ $\mathcal{T}_j \pmod 4$ if $i \pmod 2 = 1$ $\mathcal{X}$ nodes as determined by $G_{\mathcal{E},\mathcal{L}}$
$\mathcal{L}_{i,j}$	$\mathcal{E}$ nodes as determined by $G_{\mathcal{E},\mathcal{L}}$ $\mathcal{S}_i \pmod 2$ $\mathcal{A}_i \pmod 4$ $\mathcal{T}_i \pmod 4$

Node	Neighbors
$\mathcal{S}_0$	$\mathcal{C}_i$ for all i $\mathcal{X}_i^j$, where $j \in \{0,1,2\}$ and $i \in \{0,1,2\}$ every $\mathcal{E}_{i,j}$ with $i \pmod 2 = 0$ every $\mathcal{L}_{i,j}$ with $i \pmod 2 = 0$
$\mathcal{S}_1$	$\mathcal{T}_i$ for all i $\mathcal{X}_i^j$, where $j \in \{0,1,2\}$ and $i \in \{3,4,5\}$ every $\mathcal{E}_{i,j}$ with $i \pmod 2 = 1$ every $\mathcal{L}_{i,j}$ with $i \pmod 2 = 1$
$\mathcal{A}_i$	$\mathcal{T}_j$ for all j $\mathcal{R}_0$ $\mathcal{D}_j$ for all j $\mathcal{E}_{j,h}$ for all $h \pmod 4 = i$ $\mathcal{L}_{j,h}$ for all $j \pmod 4 = i$
$\mathcal{T}_i$	$\mathcal{S}_1$ $\mathcal{A}_j$ for all j $\mathcal{T}_j$ for all $j \neq i$ $\mathcal{D}_j$ for all j $\mathcal{P}$ $\mathcal{X}_h^j$, where $j \in \{0,1,2\}$ and $h \in \{3,4,5\}$ $\mathcal{E}_{j,h}$ for all $j \pmod 2 = 1$ and $h \pmod 4 = i$ $\mathcal{L}_{j,h}$ for all $j \pmod 4 = i$

Node	Neighbors
$\mathcal{R}_i$	$\mathcal{A}_j$ for all j if $i = 0$ $\mathcal{R}_j$ for $j = i-1$ and $j = i+1$ $\mathcal{C}_j$ for all $0 \le j \le \hat{n}/3$ if $i \pmod 3 = 0$ $\mathcal{C}_j$ for all $\hat{n}/3 \le j \le 2\hat{n}/3$ if $i \pmod 3 = 1$ $\mathcal{C}_0$ and $\mathcal{C}_j$ for all $2\hat{n}/3 \le j \le \hat{n}-1$ if $i \pmod 3 = 2$ $\mathcal{P}$ $\mathcal{X}_h^j$ where $j = i \pmod 3$ and $h \in \{0,1,2,3,4,5\}$
$\mathcal{C}_i$	$\mathcal{S}_0$ $\mathcal{R}_j$ for all $j \pmod 3 = 0$ if $0 \le i \le \hat{n}/3$ $\mathcal{R}_j$ for all $j \pmod 3 = 1$ if $\hat{n}/3 \le i \le 2\hat{n}/3$ $\mathcal{R}_j$ for all $j \pmod 3 = 2$ if $i = 0$ or $2\hat{n}/3 \le i \le \hat{n}-1$ $\mathcal{C}_j$ for $j \equiv i-1 \pmod{\hat{n}}$ and $j \equiv i+1 \pmod{\hat{n}}$ $\mathcal{X}_h^j$ where $j \in \{0,1,2\}$ and $h \in \{0,1,3\}$ if $i \pmod 3 = 0$ $\mathcal{X}_h^j$ where $j \in \{0,1,2\}$ and $h \in \{0,2,3\}$ if $i \pmod 3 = 1$ $\mathcal{X}_h^j$ where $j \in \{0,1,2\}$ and $h \in \{1,2,3\}$ if $i \pmod 3 = 2$
$\mathcal{D}_i$	$\mathcal{A}_j$ for all j $\mathcal{T}_j$ for all j $\mathcal{D}_j$ for $j \equiv i-1 \pmod{\hat{n}}$ and $j \equiv i+1 \pmod{\hat{n}}$ $\mathcal{P}$ $\mathcal{X}_h^j$ where $j \in \{0,1,2\}$ and $h \in \{2,4,5\}$ if $0 \le i \le \hat{n}/3-1$ $\mathcal{X}_h^j$ where $j \in \{0,1,2\}$ and $h \in \{1,4,5\}$ if $\hat{n}/3 \le i \le 2\hat{n}/3-1$ $\mathcal{X}_h^j$ where $j \in \{0,1,2\}$ and $h \in \{0,4,5\}$ if $2\hat{n}/3 \le i \le \hat{n}-1$
$\mathcal{P}$	$\mathcal{T}_i$ for all i $\mathcal{R}_i$ for all i $\mathcal{D}_i$ for all i $\mathcal{X}_i^j$ where $j \in \{0,1,2\}$ and $i = 3$

The following lemma shows that if the cops allow the robber to go to an exit of some $\mathcal{R}$ node, then they have to go back to $\{\mathcal{S}_0, \mathcal{S}_1\}$ in order to prevent the robber from going to $\mathcal{U}$.

▶ **Lemma 8.** *Consider any $\mathcal{X}_i^j$. The only cop combination not containing $\mathcal{X}_i^j$ that covers all neighbors of $\mathcal{X}_i^j$ in $\mathcal{U}$ is $\{\mathcal{S}_0, \mathcal{S}_1\}$.*

As Lemma 8 already indicates, the cops do not want the robber to be able to go to an exit from an $\mathcal{R}$ node. The next lemma characterizes the cop combinations from where they can prevent the robber from doing that.

▶ **Lemma 9.** *Consider any $\mathcal{R}_i$. The only cop combinations not containing any $\mathcal{R}_j$ with $j \equiv i \pmod 3$ that cover all exits of $\mathcal{R}_i$ are $\{\mathcal{S}_0, \mathcal{S}_1\}$, $\{\mathcal{S}_0, \mathcal{T}_j\}$ for any j, and $\{\mathcal{C}_j, \mathcal{D}_h\}$ for any pair (j,h) satisfying one of the following three conditions:*

1. $j \pmod 3 = 0$ *and* $0 \le h \le \hat{n}/3 - 1$,
2. $j \pmod 3 = 1$ *and* $\hat{n}/3 \le h \le 2\hat{n}/3 - 1$,
3. $j \pmod 3 = 2$ *and* $2\hat{n}/3 \le h \le \hat{n} - 1$.

Observe that the cop combinations from Lemma 9 are independent of the choice of the considered $\mathcal{R}_i$ which implies that these cop combinations cover all nodes in $\mathcal{X}$. We call such

Table 2 The robber's strategy.

$\{c_0(t), c_1(t)\}$	$r(t-1)$	$r(t)$
$\neq \{\mathcal{S}_0, \mathcal{S}_1\}$	some node in $\mathcal{U}$	some uncovered node in $\mathcal{U}$
$\{\mathcal{S}_0, \mathcal{S}_1\}$	some node in $\mathcal{U}$	some node in $\mathcal{A}$
$\{\mathcal{S}_0, \mathcal{S}_1\}$ or $\{\mathcal{S}_0, \mathcal{T}_i\}$	$\mathcal{A}_i$	$\mathcal{R}_0$
$\neq \{\mathcal{S}_0, \mathcal{S}_1\}$ and $\neq \{\mathcal{S}_0, \mathcal{T}_i\}$	$\mathcal{A}_i$	some uncovered node in $\mathcal{U}$
not covering all exits of $\mathcal{R}_i$	$\mathcal{R}_i$	some uncovered exit of $\mathcal{R}_i$
covering all exits of $\mathcal{R}_i$	$\mathcal{R}_i$	the uncovered node from $\{\mathcal{R}_{i-1}, \mathcal{R}_i, \mathcal{R}_{i+1}\}$ with smallest absolute index; if all are covered, stay in $\mathcal{R}_i$
$\neq \{\mathcal{S}_0, \mathcal{S}_1\}$	$\mathcal{X}_i^j$	some uncovered node in $\mathcal{U}$
$\{\mathcal{S}_0, \mathcal{S}_1\}$	$\mathcal{X}_i^j$	some node from $\{\mathcal{R}_{-1}, \mathcal{R}_0, \mathcal{R}_1\}$

a cop combination *exit-blocking*. Furthermore, we call an exit-blocking cop combination *proper* if it does not contain a node from $\mathcal{S}$ (i.e., it consists of a node from $\mathcal{C}$ and a node from $\mathcal{D}$). Lastly, we call a proper exit-blocking cop combination $\{\mathcal{C}_i, \mathcal{D}_j\}$ *forcing* if there exist $h, h' \in \{-1, 0, 1\}$, $h \neq h'$, such that the cops cover all $\mathcal{R}_{h \pmod 3}$ and all $\mathcal{R}_{h' \pmod 3}$. A close look at the construction of $G_{\hat{n}}$ shows that a proper exit-blocking cop combination $\{\mathcal{C}_i, \mathcal{D}_j\}$ is forcing if and only if $i \in \{0, \hat{n}/3, 2\hat{n}/3\}$.

Proper exit-blocking cop combinations prevent the robber from going back (too much) towards the middle of the $\mathcal{R}$ path since they contain a $\mathcal{C}$ node which by its nature is connected to every third $\mathcal{R}$ node. Thus, in order to be able to chase the robber towards one end of the $\mathcal{R}$ path, the cops have to stay in proper exit-blocking cop combinations.

The $\mathcal{C}$ node in a forcing proper exit-covering cop combination covers more $\mathcal{R}$ nodes than the $\mathcal{C}$ node in a usual proper exit-covering cop combination and thereby forces the robber to move one step towards the end of her $\mathcal{R}$ path. In order to chase the robber another step, the cops have to go to a forcing proper exit-covering cop combination containing a different $\mathcal{C}$ node. The following lemma shows a lower bound on the time it takes the cops to go from one forcing proper exit-covering cop combination to another with a different $\mathcal{C}$ node, while using only proper exit-covering cop combinations on the way. Refer to Figures 2 and 3 for illustrations of the underlying idea.

▶ **Lemma 10.** *Let* $(\{\mathcal{C}_i, \mathcal{D}_j\} = \{c_0(t), c_1(t)\}, \{c_0(t+1), c_1(t+1)\}, \ldots, \{c_0(t+h), c_1(t+h)\} = \{\mathcal{C}_{i'}, \mathcal{D}_{j'}\})$ *be a sequence of proper exit-blocking cop combinations describing the combined movement of the two cops from time* t *to time* $t+h$*. If* $\{\mathcal{C}_i, \mathcal{D}_j\}$ *and* $\{\mathcal{C}_{i'}, \mathcal{D}_{j'}\}$ *are forcing and* $i \neq i'$*, then* $h \geq \hat{n}/3 \cdot (\hat{n}/3 - 1) \in \Omega(\hat{n}^2)$.

2.4 The Robber's Strategy

Here, we explicitly specify a strategy for the robber that ensures that 2 cops need time $\Omega(n^3)$ to capture the robber in $G_{\hat{n}}$:

If the cops are in $\mathcal{S}_0$ and $\mathcal{S}_1$ in round 0, then the robber starts in $\mathcal{R}_0$, otherwise the robber starts in some node in $\mathcal{U}$ that is not covered by any of the cops (which exists by Lemma 6). Depending on where the cops are, the robber moves as specified in Table 2 (as long as she is not captured yet).

We show now that the specified strategy is well-defined, i.e., that the robber can perform any step in the strategy and that no other situations than the specified ones can occur if the robber follows the strategy. For the first part, we go through the table line by line:

Table 3 The cops' strategy.

$r(t-1)$	$(c_0(t-1), c_1(t-1))$	$(c_0(t), c_1(t))$
$\mathcal{A}_i$	$(\mathcal{S}_0, \mathcal{S}_1)$	$(\mathcal{S}_0, \mathcal{T}_i)$
$\neq \mathcal{A}_i$ for all i	$(\mathcal{S}_0, \mathcal{S}_1)$	$(\mathcal{S}_0, \mathcal{T}_j)$ for some j
arbitrary	$(\mathcal{S}_0, \mathcal{T}_i)$	$(\mathcal{C}_0, \mathcal{D}_0)$
$\neq \mathcal{C}_h$ for all h	$(\mathcal{C}_i, \mathcal{D}_j)$	$\left(\mathcal{C}_{i+1 \pmod{\hat{n}}}, \mathcal{D}_{j+1 \pmod{\hat{n}}}\right)$ if this covers all nodes in $\mathcal{X}$ $\left(\mathcal{C}_i, \mathcal{D}_{j+1 \pmod{\hat{n}}}\right)$ otherwise
$\mathcal{C}_h$	$(\mathcal{C}_i, \mathcal{D}_j)$	$(\mathcal{S}_0, \mathcal{P})$

By Lemma 6, if the robber is in some node $u \in \mathcal{U}$, then she can always go to some uncovered node in $\mathcal{U}$, provided the cops are not in $\mathcal{S}_0$ and $\mathcal{S}_1$. She can also go from u to some node in $\mathcal{A}$ since any node in $\mathcal{U}$ has some node in $\mathcal{A}$ as a neighbor, by the construction of $G_{\hat{n}}$. Similarly, the robber can go from any node in $\mathcal{A}$ to $\mathcal{R}_0$. By Lemma 7, if the robber is in some node $\mathcal{A}_i$, then she can always go to some uncovered node in $\mathcal{U}$, provided the cops are not in $\mathcal{S}_0$ and $\mathcal{S}_1$ or in $\mathcal{S}_0$ and $\mathcal{T}_i$. The instructions where to go to from $\mathcal{R}_i$ are trivially satisfiable. From $\mathcal{X}_i^j$, the robber can always go to some uncovered node in $\mathcal{U}$ if the cops are not in $\mathcal{S}_0$ and $\mathcal{S}_1$, by Lemma 8. She can also go to either $\mathcal{R}_{-1}$, $\mathcal{R}_0$ or $\mathcal{R}_1$ from $\mathcal{X}_i^j$ since each $\mathcal{X}_i^j$ is connected to exactly one of those three $\mathcal{R}$ nodes, by the construction of $G_{\hat{n}}$.

Moreover since the robber starts in a node in $\mathcal{U}$ or $\mathcal{R}$, Table 2 covers all situations where the robber is in some node in $\mathcal{U}$, $\mathcal{A}$, $\mathcal{R}$ or $\mathcal{X}$, and each instruction ends with the robber being in one of those nodes, the presented strategy specifies what the robber has to do for every possibly occurring situation.

2.5 The Cops' Strategy

Now, we explicitly specify a strategy for the cops that ensures that the robber is captured in time $\mathcal{O}(n^3)$ in $G_{\hat{n}}$:

Cop 0 starts in $\mathcal{S}_0$ and Cop 1 starts in $\mathcal{S}_1$ in round 0. Depending on where the robber is, the cops move as specified in Table 3. There is one exception however: If a cop can capture the robber immediately, then she does so, overriding any possible instruction from the table.

We show now that the specified strategy is well-defined, i.e., that the cops can actually perform any step in the strategy and that no other situations than the specified ones can actually occur[2] if the cops follow the strategy:

The construction of $G_{\hat{n}}$ ensures that the cops can actually move from the cop combinations at time $t-1$ given in Table 3 to the cop combinations at time t. Since the cops start in $\mathcal{S}_0$ and $\mathcal{S}_1$, the only thing that is left to show is that from $(\mathcal{S}_0, \mathcal{P})$ (which is the only output combination that is not dealt with on the input side) the cops can capture the robber at time $t+1$, provided that the robber is in some $\mathcal{C}_h$ at time $t-1$. For that, it is sufficient to observe that any neighbor of $\mathcal{C}_h$, and $\mathcal{C}_h$ itself, is covered by $\mathcal{S}_0$ or $\mathcal{P}$.

2.6 A Lower Bound for the Robber's Strategy

Here, we show that the strategy for the robber specified in Table 2 ensures that the cops need time $\Omega(n^3)$ to capture the robber in $G_{\hat{n}}$. For convenience, we assume throughout the

[2] More precisely, if a situation occurs that is not specified in Table 3, then the cops can capture the robber immediately.

following lower bound considerations that if a cop can capture the robber immediately, then she does so. This certainly cannot worsen any strategy the cops follow.

We start by observing that the set of $\mathcal{R}$ nodes can be partitioned into three roughly equally-sized sets such that the $\mathcal{R}$ nodes in each such set have exactly the same exits. As the following lemma shows, if the robber is in an $\mathcal{R}$ node, then she does not need to worry about a cop being in another $\mathcal{R}$ node that has (and therefore covers) the same exits, since such a situation cannot occur if the robber follows the specified strategy.

We proceed by determining the nodes the robber can be captured in. Then, using Lemma 11 and Lemma 12, we give a lower bound on the capture time of $G_{\hat{n}}$.

▶ **Lemma 11.** *If the robber follows the strategy specified in Section 2.4, then the following holds: If the robber is in some node $\mathcal{R}_i$ at time t and is not captured at time $t+1$, then neither of the 2 cops can be in some node $\mathcal{R}_j$ with $j \equiv i \pmod 3$ at time $t+1$.*

▶ **Lemma 12.** *If the robber follows the strategy specified in Section 2.4, then she can only be captured in $\mathcal{R}_{\hat{n}}$ or $\mathcal{R}_{-\hat{n}}$.*

▶ **Lemma 13.** *If the robber follows the strategy specified in Section 2.4, then 2 cops need time $\Omega(\hat{n}^3)$ to capture the robber in $G_{\hat{n}}$.*

2.7 An Upper Bound for the Cops' Strategy

While the aim of this work is a lower bound, we need to show that 2 cops can actually capture the robber in $G_{\hat{n}}$, in order to use $G_{\hat{n}}$ as a lower bound graph for the capture time for 2 cops in 2-cop-win graphs. We start by showing that from a proper exit-blocking cop combination the cops can always go to another proper exit-blocking cop combination by doing one of the following: Both cops move to the next node in their respective cycle or only the cop in the $\mathcal{D}$ cycle moves to the next node.

▶ **Lemma 14.** *If $(\mathcal{C}_i, \mathcal{D}_j)$ is an exit-blocking cop combination, then it holds that at least one of $(\mathcal{C}_i, \mathcal{D}_{j+1 \pmod{\hat{n}}})$ and $(\mathcal{C}_{i+1 \pmod{\hat{n}}}, \mathcal{D}_{j+1 \pmod{\hat{n}}})$ is an exit-blocking cop combination.*

The following lemma shows that, once the cops reach $\mathcal{C}_0$ and $\mathcal{D}_0$, the robber cannot ever leave $\mathcal{R}$ without being captured in the next two moves. Then, using Lemma 14 and Lemma 15, we give an upper bound on the capture time of $G_{\hat{n}}$.

▶ **Lemma 15.** *Let $r(t) \in \mathcal{R}$ and $(c_0(t+1), c_1(t+1)) = (\mathcal{C}_0, \mathcal{D}_0)$ for some point in time t. If the robber leaves $\mathcal{R}$ at some later point in time t', i.e., if $r(t') \notin \mathcal{R}$ for some $t' > t$, then the robber will be captured at time $t'' \leq t' + 2$, provided the two cops follow the strategy specified in Section 2.5.*

▶ **Lemma 16.** *If the two cops follow the strategy specified in Section 2.5, then they capture the robber in time $\mathcal{O}(\hat{n}^3)$ in $G_{\hat{n}}$.*

Finally, by Lemma 13 we get that for the case of 2 cops, the capture time of the graph family $\{G_{\hat{n}}\}_{\hat{n} \geq 3, \hat{n} \equiv 0 \pmod 3}$ is $\Omega(\hat{n}^3) \subseteq \Omega(n^3)$ and by Lemma 16 we get that every graph in $\{G_{\hat{n}}\}_{\hat{n} \geq 3, \hat{n} \equiv 0 \pmod 3}$ is 2-cop-win. Together, these lemmas yield Theorem 1 for 2 cops.

The Case of $k > 2$

Our graph construction and the corresponding lower bound proofs follow closely the design of the case of two cops. To accommodate a third cop, Cop 2, we essentially copy the $\mathcal{D}$ component and ensure, that for every step of Cop 1, Cop 2 has to perform $\Omega(\hat{n})$ steps. For the case of $k > 3$ cops, we simply apply this trick inductively. Due to space limitations, we defer the detailed discussion of the case of $k > 2$ cops to the full version.

References

1 I. Abraham, C. Gavoille, A. Gupta, O. Neiman, and K. Talwar. Cops, robbers, and threatening skeletons: padded decomposition for minor-free graphs. In *Proccedings of the 46th ACM Symposium on Theory of Computing, STOC*, pages 79–88, 2014.

2 M. Aigner and M. Fromme. A Game of Cops and Robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984. doi:10.1016/0166-218X(84)90073-8.

3 B. Alspach. Searching and Sweeping Graphs: a Brief Survey. *Le Matematiche*, 59:5–37, 2006.

4 T. Andreae. On a pursuit game played on graphs for which a minor is excluded. *Journal of Combinatorial Theory, Series B*, 41(1):37–47, 1986.

5 A. Berarducci and B. Intrigila. On the Cop Number of a Graph. *Advances in Applied Mathematics*, 14(4):389–403, 1993. doi:10.1006/aama.1993.1019.

6 A. Bonato and W. Baird. Meyniel's Conjecture on the Cop Number: a Survey. *Journal of Combinatorics*, 3:225–238, 2012.

7 A. Bonato, P. A. Golovach, G. Hahn, and J. Kratochvíl. The Capture Time of a Graph. *Discrete Mathematics*, 309(18):5588–5595, 2009. doi:10.1016/j.disc.2008.04.004.

8 A. Bonato, P. Gordinowicz, B. Kinnersley, and P. Pralat. The Capture Time of the Hypercube. *Electr. J. Comb.*, 20(2):P24, 2013.

9 A. Bonato and R. J. Nowakowski. *The Game of Cops and Robbers on Graphs*, volume 61 of *Student Mathematical Library*. American Mathematical Society, Providence, RI, 2011.

10 A. Bonato and B. Yang. Graph Searching and Related Problems. In *Handbook of Combinatorial Optimization*, pages 1511–1558. Springer New York, 2013. doi:10.1007/978-1-4419-7997-1_76.

11 N. E. Clarke and G. MacGillivray. Characterizations of k-copwin Graphs. *Discrete Mathematics*, 312(8):1421–1425, 2012. doi:10.1016/j.disc.2012.01.002.

12 F. V. Fomin and D. M. Thilikos. An Annotated Bibliography on Guaranteed Graph Searching. *Theor. Comput. Sci.*, 399(3):236–245, 2008.

13 K.-T. Förster, R. Nuridini, J. Uitto, and R. Wattenhofer. Lower Bounds for the Capture Time: Linear, Quadratic, and Beyond. In *22nd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 342–356, 2015.

14 P. Frankl. Cops and Robbers in Graphs with Large Girth and Cayley Graphs. *Discrete Appl. Math.*, 17(3):301–305, June 1987. doi:10.1016/0166-218X(87)90033-3.

15 G. Hahn. Cops, Robbers and Graphs. *Tatra Mt. Math. Publ*, 36(163):163–176, 2007.

16 A. Kehagias and P. Pralat. Some Remarks on Cops and Drunk Robbers. *Theoretical Computer Science*, 463:133–147, 2012.

17 L. Lu and X. Peng. On Meyniel's Conjecture of the Cop Number. *Journal of Graph Theory*, 71(2):192–205, 2012. doi:10.1002/jgt.20642.

18 A. Mehrabian. The Capture Time of Grids. *Discrete Mathematics*, 311(1):102–105, 2011. doi:10.1016/j.disc.2010.10.002.

19 R. J. Nowakowski and P. Winkler. Vertex-to-vertex Pursuit in a Graph. *Discrete Mathematics*, 43(2-3):235–239, 1983. doi:10.1016/0012-365X(83)90160-7.

20 P. Pralat. When Does a Random Graph Have a Constant Cop Number. *Australasian Journal of Combinatorics*, 46:285–296, 2010.

21 A. Quilliot. *Jeux et Pointes Fixes sur les Graphes*. PhD thesis, Universite de Paris VI, 1978.

Stochastic Control via Entropy Compression*

Dimitris Achlioptas[†1], **Fotis Iliopoulos**[‡2], **and Nikos Vlassis**[3]

1 **Department of Computer Science, UC Santa Cruz, Santa Cruz, CA, USA**
optas@cs.ucsc.edu
2 **Department of Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA, USA**
fotis.iliopoulos@berkeley.edu
3 **Adobe Research, San Jose, CA, USA**
nikos.vlassis@gmail.com

Abstract

Consider an agent trying to bring a system to an acceptable state by repeated probabilistic action. Several recent works on algorithmizations of the Lovász Local Lemma (LLL) can be seen as establishing sufficient conditions for the agent to succeed. Here we study whether such stochastic control is also possible in a noisy environment, where both the process of state-observation and the process of state-evolution are subject to adversarial perturbation (noise). The introduction of noise causes the tools developed for LLL algorithmization to break down since the key LLL ingredient, the sparsity of the causality (dependence) relationship, no longer holds. To overcome this challenge we develop a new analysis where entropy plays a central role, both to measure the rate at which progress towards an acceptable state is made and the rate at which noise undoes this progress. The end result is a sufficient condition that allows a smooth tradeoff between the intensity of the noise and the amenability of the system, recovering an asymmetric LLL condition in the noiseless case.

1998 ACM Subject Classification G.3 Probabilistic Algorithms, I.2.8 Control Theory

Keywords and phrases Stochastic Control, Lovász Local Lemma

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.83

1 Introduction

Consider a system with a large state space Ω, hidden from view inside a box. On the outside of the box there are lightbulbs and buttons. Each lightbulb corresponds to a set $f_i \subseteq \Omega$ and is lit whenever the current state of the system is in f_i. We think of each set f_i as containing all states sharing some negative feature $i \in [m]$ and refer to each such set as a *flaw*, letting $F = \{f_1, f_2, \ldots, f_m\}$. For example, if the system corresponds to a graph G with n vertices each of which can take one of q colors, then $\Omega = [q]^n$, and we can define for each edge e_i of G the flaw f_i to contain all assignments of colors to the vertices of G that assign the same color to the endpoints of e_i. Following linguistic convention, instead of mathematical, we will say that flaw f is present in state σ whenever $f \ni \sigma$ and that state σ is *flawless* if no flaw is present in σ. The buttons correspond to actions, i.e., to mechanisms for state evolution. Specifically, taking action a while in state σ moves the system to a new state τ, selected from a probability distribution that depends on both σ and a.

* A full version of this paper appears as [1], https://arxiv.org/abs/1607.06494.
† Research supported by NSF grant CCF-1514128.
‡ Research supported by NSF grant CCF-1514434. Part of of this work was done while at Adobe Research.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 83; pp. 83:1–83:13


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

Outside the box, an agent called the *controller* observes the lightbulbs and pushes buttons, in an effort to bring the system to a flawless state. Specifically, if $O(\sigma) \in \{0,1\}^m$ denotes the lightbulb bitvector, with 1 corresponding to lit, the controller repeatedly applies a function P, called a *policy*, that maps $O(\sigma)$ to a distribution over actions. Thus, overall, state evolution proceeds as follows: if the current (hidden) state is $\sigma \in \Omega$, the controller observes $O(\sigma)$ and samples an action from $P(O(\sigma))$; after she takes the chosen action, the system, internally and probabilistically, moves to a new (hidden) state τ, selected from a distribution that depends on both σ and the action taken.

Our work begins with the observation that several recent results [23, 24, 18, 14, 3, 15, 2, 19] on LLL algorithmization can be seen as giving sufficient conditions for a controller as above to be able to bring the system to a flawless state quickly, with high probability. Motivated by this viewpoint we ask if conditions for LLL algorithmizations can be seen as *stability criteria* and give results for more general settings, e.g., Partially Observable Markov Decision Processes (POMDPs). Given the capacity of LLL algorithmization arguments to establish convergence in highly non-convex domains, a major pain point in control theory, we believe that bringing such arguments to stochastic control is a first step in a fruitful direction. In order to move in that direction we generalize the setting described so far in two ways:

- The mapping O from states to observations is *stochastic*: the lightbulbs are unreliable, exhibiting both false-positives and false-negatives.
- Both the environment surrounding the system and the implementation of actions are *noisy*: the controller is not the only agent affecting state evolution and flaws may be introduced into the state for reasons unrelated to her actions, even spontaneously.

Naturally, the question is whether sufficient conditions for quick convergence to flawless states can still be established in this setting. We answer the question affirmatively and show, in a precise mathematical sense, that the less internal conflict there is in the system, the more noise the controller can tolerate. In order to prove this we require the controller to be *focused* and to *prioritize*. That is, we will assume that the flaws are ordered by priority according to an arbitrary but fixed permutation π of F, and we will ascribe the action taken by the controller in each step to the present flaw (focus) of highest priority (prioritization). The analysis will then take into account both how good the actions are at ridding the state of that flaw and how damaging they are in terms of introducing new flaws. In particular, with this attribution mechanism in place, and similarly to LLL algorithmization arguments, we will say that flaw f_i can cause flaw f_j if there exists a state transition with non-zero probability under the policy, from a state in which f_i is the highest priority flaw and f_j is absent, to a state in which f_j is present.

The main challenge we face is that in the presence of noise the causality relationship becomes dense. To overcome this we develop a new analysis in which causality is not a binary relationship, but one weighted by the *frequency* of interactions. In particular, our condition guaranteeing that the controller will succeed within a reasonable amount of time allows the causality graph to become arbitrarily dense, if the frequency of interactions is sufficiently small. Turning the sparsity of the causality relationship into a *soft* requirement is a major departure from the LLL setting and our main technical contribution. We do this by developing an entropy compression argument, in which we carefully amortize the entropy injected into the system to encode the effect of noise on the state trajectory. It is worth pointing out that even though our technique applies to the far more general noisy setting, in the absence of noise it recovers the main result of [3], thus providing a smooth relationship between lack of internal conflict and robustness to noise.

2 Formal Setting and Statement of Results

In the absence of observational and environmental noise we can think of the state evolution under a policy P as a random walk on a certain digraph on Ω. Specifically, at each flawed state $\sigma \in \Omega$, for each action in the support of $P(O(\sigma))$, there is a bundle of outgoing arcs of total probability 1, corresponding to the state-transitions from σ under this action. The convolution of $P(O(\sigma))$ with the distribution inside each bundle yields the state-transition probability distribution from each flawed state σ.

The presence of observational and environmental noise both distorts the transition probabilities and introduces new transitions. For example, whenever observational noise causes $O(\sigma)$ to differ from the set of flaws truly present in σ, the controller may chose an action (from the support of $P(O(\sigma))$) under which there are transitions from σ that were not present in the noise-free digraph. We model the overall distortion induced by noise by taking the noise-free digraph, which we think of as the *principal* mechanism for state evolution, reducing the probabilities on all its edges uniformly by a factor of $1-p$, and allowing the leftover probability mass to be distributed arbitrarily, in order to form the noise. More precisely:

- Let D_{po} be the digraph on Ω of possible state-transitions under policy P, with a self-loop added at every flawless state. Let ρ_{po} be the P-induced state-transition probability distribution, augmented so that all self-loops at flawless states have probability 1.
- Let D_{ns} be an **arbitrary** digraph on Ω. For each vertex σ in D_{ns}, let $\rho_{\text{ns}}(\sigma,\cdot)$ be an **arbitrary** probability distribution on the arcs leaving σ.
- We will analyze the Markov chain on Ω which at every $\sigma \in \Omega$, with probability p follows an arc in D_{ns} and with probability $1-p$ follows an arc in D_{po}. Formally, for every $\sigma \in \Omega$,

$$\rho(\sigma,\cdot) = (1-p)\cdot\rho_{\text{po}}(\sigma,\cdot) + p\cdot\rho_{\text{ns}}(\sigma,\cdot) \ .$$

  We assume that the system starts at a state σ_1, according to some unknown probability distribution θ.

Requiring that the effect of noise is captured by a mixture of the original (principal) chain and an arbitrary chain is the only assumption that we make. In particular, by allowing D_{ns} and ρ_{ns} to be arbitrary we forego the need to posit specific models of observational and environmental noise, lending greater generality to our results. To see this, let $U(\sigma)$ denote the set of flaws actually present in σ (and, slightly abusing notation, also the characteristic vector of $U(\sigma) \subseteq F$). In any step where the state transition distribution is not the principal one, we can think of this as occurring because $O(\sigma) \neq U(\sigma)$ and the distribution corresponds to $P(O(\sigma))$, or because $O(\sigma) \neq U(\sigma)$ and the distribution does not *even* correspond to $P(O(\sigma))$, or because $O(\sigma) = U(\sigma)$ but, silently, the distribution followed is different from $P(O(\sigma))$. In particular, notice that whenever $O(\sigma) = \mathbf{0}$, the controller thinks she has arrived at a flawless state and, thus, authorizes a self-loop with probability 1. In such a case, the fact that the system will follow ρ_{ns} with probability p means that we are allowing the noise not only to trick the controller to inactivity but also to silently move the system to a new state. Similarly, after the system arrives at a flawless state, i.e., $U(\sigma) = \mathbf{0}$, with probability p it will then follow an arc in D_{ns}, potentially to a flawed state. We allow this to occur to be consistent with (i) the idea that observational noise can occur at any state, even a flawless one, thus causing unneeded, potentially detrimental action, and (ii) with the idea that flaws can be introduced spontaneously from the environment at any state. Our goal is, thus, to prove that from *any* initial state, after a small number of steps, the system will

reach a flawless state, despite the noise. As we will see, what will matter about the noise is the extent to which noise-induced transitions introduce flaws in the state.

Let $D = D_{\rm po} \cup D_{\rm ns}$. To avoid certain trivialities we will assume that there exists a constant $B < \infty$ such that $2^{-B} < \rho(\sigma, \tau) < 1 - 2^{-B}$ for every arc $(\sigma, \tau) \in D$. For each state σ, we denote the highest priority flaw present in σ by $\pi(\sigma)$; if $\pi(\sigma) = f_i$, we label all arcs leaving σ as $\sigma \xrightarrow{i} \cdot$, i.e., with the index of the flaw to which we attribute the transition (we use i instead of f_i as the label to lighten notation). We will refer to $\pi(\sigma)$ as the flaw *addressed* at σ.

▶ **Causality.** *For an arc $\sigma \xrightarrow{i} \tau$ in D and a flaw f_j present in τ we say that f_i causes f_j if $f_j \not\ni \sigma$. The digraph on $[m]$ where $i \to j$ iff D contains an arc such that f_i causes f_j is the causality digraph $C(D)$.*

▶ **Neighborhood.** *The* neighborhood *of a flaw f_i in $C = C(D)$ is $\Gamma(f_i) = \{f_i\} \cup \{f_j : i \to j$ exists in $C\}$.*

For our condition we will need to bound from *below* the entropy injected into the system in each step. To that end we define the potential of each flaw f_i to be

$$\text{Potential}(f_i) = \min_{\sigma:\pi(\sigma)=f_i} H[\rho(\sigma, \cdot)] \ . \tag{1}$$

We extend the definition to sets of flaws i.e., $\text{Potential}(S) = \sum_{f \in S} \text{Potential}(f)$, where $\text{Potential}(\emptyset) = 0$.

In the absence of noise, $\text{Potential}(f_i)$ expresses a lower bound on the diversity of ways to address flaw f_i, by bounding from below the "average number of random bits consumed" whenever f_i is addressed. Thus, it bounds from below the rate at which the controller explores the state space *locally*. The presence of noise may decrease or may increase the potential. For example, if all arcs in $D_{\rm ns}$ are self-loops, then the noise is equivalent to the action-buttons "sometimes not working" and its only (and very benign) effect is to slow down the exploration by a constant factor. At the other extreme, if $D_{\rm ns}$ is the complete digraph on Ω and $\rho_{\rm ns}$ is uniform, then (unless p is extremely small) the situation is, clearly, hopeless. Correspondingly, even though the potential has been greatly increased, the causality relationship is complete. We note that, trivially, the potential of each flaw is bounded from below by the minimum entropy injected by the principal alone whenever the flaw is addressed, i.e., $\text{Potential}(f_i) \geq (1-p) \min_{\sigma:\pi(\sigma)=f_i} H[\rho_{\rm po}(\sigma)]$.

The other important characteristic of each flaw f_i is its congestion, i.e., the maximum number of arcs with label i that lead to the same state. For the same reason we would like the potential of a flaw to be big, we would like its congestion to be small: if arcs from different states in f_i lead to the same state, then exploration slows down and the entropy injected into the system must be appropriately discounted in order to yield a good measure of the rate of state space exploration. To see this observe that $\text{Potential}(f_i)$ is independent of the destinations of the arcs leaving f_i and compare the case where these destinations are all distinct with the case where they all lie in a small (bottleneck) set. As the congestion due to the principal and the congestion due to noise will have different effects, we need to account for them separately. Let $A_{\rm po}(\sigma)$ denote the support of $\rho_{\rm po}(\sigma, \cdot)$ and $A_{\rm ns}(\sigma)$ denote the support of $\rho_{\rm ns}(\sigma, \cdot)$.

▶ **Congestion.** *For any flaw $f_i \in F$, let*

$$\begin{aligned} \text{Congestion}_{\rm po}(f_i) &= \max_{\tau \in \Omega} |\{\sigma \in f_i : \tau \in A_{\rm po}(\sigma)\}| \\ \text{Congestion}_{\rm ns}(f_i) &= \max_{\tau \in \Omega} |\{\sigma \in f_i : \tau \in A_{\rm ns}(\sigma)\}| \ . \end{aligned}$$

Let $\mathrm{b}_{\mathrm{po}}^{f_i} = \log_2 \mathrm{Congestion}_{\mathrm{po}}(f_i)$. *Let* $\mathrm{b}_{\mathrm{ns}}^{f_i} = \log_2 \mathrm{Congestion}_{\mathrm{ns}}(f_i)$. *Let* $\mathrm{b}_{\mathrm{ns}} = \max_{f_i \in F} \mathrm{b}_{\mathrm{ns}}^{f_i}$.

Let C_{po} and C_{ns} be the causality graphs of D_{po} and D_{ns}, respectively, and let $\Gamma_{\mathrm{po}}(f_i)$ and $\Gamma_{\mathrm{ns}}(f_i)$ be the corresponding neighborhoords. Let $\Delta_i = |\Gamma_{\mathrm{ns}}(f_i)|$. Recall that $h(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the *binary entropy* of $p \in [0,1]$. To express the lost efficiency due to noise in addressing flaw f_i, we let

$$\begin{aligned} q_i(p) &= p\left(\Delta_i\left(\mathrm{b}_{\mathrm{ns}} + \frac{5}{2} + h(p)\right) - 2 - h(p)\right) \\ &\leq p\,\Delta_i(\mathrm{b}_{\mathrm{ns}} + 4) \ . \end{aligned}$$

Observe that $q_i(p)$ is independent of the policy and that its leading term is $p\Delta_i$. This means that, unlike the LLL, the number, Δ_i, of different flaws that may be introduced when addressing a flaw can be arbitrarily large if the *frequency of interactions* between flaws, captured by p, is sufficiently small. Our main result establishes a condition under which the probability of not reaching a flawless state within $O(\log_2|\Omega| + m)$ steps is exponentially small. To state it define for each flaw f_i,

$$\mathrm{Amenability}(f_i) = \mathrm{Potential}(f_i) - \mathrm{b}_{\mathrm{po}}^{f_i} \ .$$

▶ **Theorem 1.** *If for every flaw* $f_i \in F$,

$$\sum_{f_j \in \Gamma_{\mathrm{po}}(f_i)} 2^{-\mathrm{Amenability}(f_j) + q_j(p)} < 2^{-(2+h(p))} \ , \tag{2}$$

then there exists a constant $R > 0$ *depending on the slack in* (2)*, such that for every* $s > 1/2$*, the probability of not reaching a flawless state after* $Rs(\log_2|\Omega| + m)$ *steps is less than* $\exp(-s)$.

▶ Remark. In the noiseless case, i.e., when $p = 0$, equation (2) becomes an asymmetric LLL criterion. In particular, the main result of [3] is that if $\mathrm{b}_{\mathrm{po}}^{f_i} = 0$ and all distributions $\rho_{\mathrm{po}}(\sigma, \cdot)$ are uniform over their support $A_{\mathrm{po}}(\sigma)$, then, a sufficient condition for reaching a flawless state quickly is that for every $f_i \in F$, $\sum_{f_j \in \Gamma_{\mathrm{po}}(f_i)} 1/a_j < 1/\mathrm{e}$, where $a_j = \min_{\sigma \in f_j : \pi(\sigma) = f_j} |A_{\mathrm{po}}(\sigma)|$. We see that in this setting our condition (2) recovers this, up to the constant on the right hand side, i.e., $1/4$ vs. $1/\mathrm{e}$.

3 Related Work

3.1 POMDPs and the Reachability Problem

Markov Decision Processes (MDPs) are widely used models for describing problems in stochastic dynamical systems [13, 28, 7], where an agent repeatedly takes actions to achieve a specific goal while the environment reacts to these actions in a stochastic way. In an MDP the agent is assumed to be able to *perfecty* observe the current state of the system and take action based on her observations. In a *partially* observable Markov Decision Process (POMDP) the agent only receives limited, and possibly inaccurate, information about the current state of the system. POMDPs have been used to model and analyze problems in artificial intelligence and machine learning such as reinforcement learning [9, 17], planning under uncertainty [16], etc.

Formally, a discrete POMDP is defined by the following primitives (all sets are assumed finite): (i) a *state space* Ω, (ii) a finite alphabet of *actions* $\mathcal{A}$, (iii) an observation space $\mathcal{O}$, (iv) an *action*-conditioned state transition model $\Pr(\tau|\sigma, a)$, where $\sigma, \tau \in \Omega$ and $a \in \mathcal{A}$,

(v) an observation model $\Pr(o|\sigma)$, where $\sigma \in \Omega$ and $o \in \mathcal{O}$, (vi) a cost function $\mathrm{c} : \Omega \mapsto \mathbb{R}$ (or more generally a map from state-action pairs to the reals), and (vii) a desired criterion to minimize, e.g., expected total cumulative cost $\sum_{t=0}^{\infty} \mathbb{E}\left[\mathrm{c}(\sigma_t)\right]$, where σ_t is the random variable that equals the t-th state of the trajectory of the agent. Finally, various choices of controllers are possible. For instance, a *stochastic memoryless* controller is a map from the *current* observation to a probability distribution over actions, whereas a *belief-based* controller conditions its actions on probability distributions over the state space (i.e., beliefs) that are sequentially updated (using Bayes rule or some approximation of it) while the agent is interacting with the environment.

Unfortunately, the problem of computing an optimal policy for a POMDP, i.e., designing a controller that minimizes the expected cost, is highly intractable [27, 25] and, in general, undecidable [21]. Notably, the problem remains hard even if we severely restrict the class of controllers over which we optimize [27, 20, 12, 31]. As far as we know, the only tractable case [31] requires both the cost function and the class of controllers over which we optimize to be extremely restricted. In particular, the controller can not observe or remember anything and must apply the same distribution over actions in every step.

An important special case that has motivated our work is the *reachability* problem for POMPDs. Here, one has a set of *target* states $T \subseteq \Omega$, and the goal is to design a controller that starting from a distribution θ over Ω, guides the agent to a state in T (almost surely) with the optimal expected total cumulative cost. As shown in [8], the problem is undecidable in the general case. In the same work, for the case where the costs are positive integers and the observation model is deterministic, i.e., the observations induce a partition of the state space, the authors give an algorithm which runs in time doubly-exponential in $|\Omega|$ and returns doubly-exponential lower and upper bounds for the optimal expected total cumulative cost, using a belief-based controller. On the other hand, our work establishes a sufficient condition for a stochastic memoryless controller to reach the target set T rapidly (in time logarithmic in $|\Omega|$ and linear in $|F|$), in the case where each individual observation is binary valued (set membership) and the observation model is arbitrarily stochastic. To our knowledge, this is the first tractability result for a nontrivial class of POMDPs under stochastic memoryless controllers.

3.2 Focusing and Prioritization

To achieve our results the controller must be focused and prioritize. The idea of focusing was introduced by Papadimitriou [26] in the context of satisfiability algorithms, and amounts to "if it ain't broken don't fix it", i.e., state evolution should only happen by changing the values of variables that participate in at least one violated constraint. One way to implement this idea is to always first select a violated constraint (flaw) and then take actions that tend to get rid of it. This has been an extremely successful idea in practice [29, 4] and it is often materialized by selecting a *random* flaw to address in each step. We remark that our methods allow, in fact, also the analysis of controllers that address a random flaw in each step, but for simplicity of exposition we chose to only present the case of a fixed permutation (prioritization).

Focusing is not only a good algorithmic idea, but also enables proofs of termination. Specifically, at the foundation of the argument of Moser and Tardos [24] is the following observation: whenever an algorithm (focused or not) takes t or more steps to reach a flawless state, say through flawed states $\sigma_1, \sigma_2, \ldots, \sigma_t$, there exists, by definition, a sequence of flaws $w_1, w_2, \ldots, w_t$ such that $\sigma_i \in w_i$. Therefore, by establishing a (potentially randomized) rule for selecting a flaw present in the state at each step, we can construct a random variable

$W_t = w_1, w_2, \ldots, w_t$ to act as a *witness* of the fact that the algorithm took at least t steps. While, though, prima facie all constructions are equivalent, our capacity to bound the set of all possible such sequences is not. In particular, if the algorithm is focused and in each step we report the flaw on which the algorithm focused, then we can take advantage of the following observation: each appearance of a flaw f_i in the witness sequence, with the potential exception of the very first, must be preceded by a distinct appearance of a flaw f_j that causes f_i. This allows us to bound the rate at which the entropy of the set of t-witness sequences grows with t. Of course, in a general setting, there is good reason to believe that prioritization, i.e., focusing on the flaw determined by a fixed permutation, will be not be the best one can do. In particular, observe that for the same $D_{\rm po}$, different permutations π give rise to different causality graphs. On the other hand, at the level of generality of this work, i.e., without any assumptions about the system at hand, we can not really hope for a more intelligent choice.

3.3 LLL algorithmization

The Lovász Local Lemma (LLL) [11] is a non-constructive method for proving the *existence* of flawless states that has served as a cornerstone of the probabilistic method. To use the LLL one provides a probability measure μ on Ω, often the uniform measure, transforming flaws to ("bad") events, so that the existence of flawless states is equivalent to $\mu(\bigcup_{i=1}^{m} f_i) < 1$. The key quantity to control in order to prove this is negative dependence, i.e., the extent to which the probability of a bad event may be increased (boosted) by conditioning on the non-occurrence of other bad events. Roughly speaking, the LLL requires that for each bad event f, only a small number of other bad events should be able to boost $\mu(f)$ in this manner, whereas conditioning on the non-occurrence of all other bad events should not increase $\mu(f)$. Representing the boosting relationship in a graphical manner, with vertices corresponding to bad events pointing to their potential boosters, at a high level, the LLL requirement is that this digraph is sparse.

As one can imagine, whenever one proves that Ω contains flawless objects via the LLL it is natural to then ask if some such object can be found efficiently. Making the LLL constructive has been a long quest, starting with the work of Beck [6], with subsequent works of Alon [5], Molloy and Reed [22], Czumaj and Scheideler [10], Srinivasan [30] and others. Each of these works established a method for finding flawless objects efficiently, but with additional conditions relative to the LLL. A breakthrough was made by Moser [23] who gave a very elegant algorithmization of the LLL for satisfiability via entropy compression. Very shortly afterwards, Moser and Tardos in a landmark paper [24] made the LLL constructive for every product measure μ. Specifically, they proved that if one starts by sampling an initial state according to μ, and in every step selects an arbitrary occurring bad event and resamples its variables according to μ, then with high probability a flawless state will be reached within $O(m)$ steps.

Following [24], several works [18, 14, 3, 15, 2, 19] have extended the scope of LLL algorithmization beyond product measures. In these works, unlike [24], one has to also provide either an explicit algorithm [18, 14], or an algorithmic framework [3, 2, 15, 19], along with a way to capture the *compatibility* between the algorithm's actions for addressing each flaw f_i and the measure μ. As was shown in [15, 2, 19], one can capture compatibility by letting

$$d_i = \max_{\tau \in \Omega} \frac{\nu_i(\tau)}{\mu(\tau)} \geq 1 \ , \tag{3}$$

where $\nu_i(\tau)$ is the probability of ending up at state τ at the end of the following experiment: sample $\sigma \in f_i$ according to μ and address flaw f_i at σ. An algorithm achieving $d_i = 1$ is a *resampling oracle* for flaw f_i. If $d_i = 1$ for every $i \in [m]$, then it was proven in [15] that the causality digraph equals the boosting digraph mentioned above and the condition for success is identical to that of the LLL (observe that the resampling algorithm of Moser and Tardos [24] is trivially a resampling oracle for every flaw). More generally, ascribing to each flaw f_i the *charge* $\gamma(f_i) = d_i \cdot \mu(f_i)$, yields the following user-friendly algorithmization condition [2], akin to the asymmetric Local Lemma: if for every flaw $f_i \in F$,

$$\sum_{f_j \in \Gamma(f_i)} \gamma(f_j) < \frac{1}{4} \ , \tag{4}$$

then with high probability the algorithm will reach a sink after $O(\log |\Omega| + m)$ steps.

Even though the noiseless case is only tangential to the main point of this work, as an indication of the sharpness of our analysis, we point out that in the noiseless case, our condition (2) is identical to (4) with $\gamma(f_i)$ replaced by $\chi(f_i) := 2^{-\text{Potential}(f_i)+b^{f_i}_{\text{po}}}$. In general, $\gamma(f_i)$ and $\chi(f_i)$ are incomparable. Roughly speaking, settings where $b^{f_i}_{\text{po}}$ is small and d_i is large favor $\chi(f_i)$ over $\gamma(f_i)$ and vice versa, while the two meet when $b^{f_i}_{\text{po}} = 0$, μ is uniform, and the transition probabilities are uniform, as in [3].

In terms of techniques, as hinted in Section 3.2, proofs of LLL algorithmizations consist of two independent parts. In one part, one bounds from above the probability of any witness sequence occurring, or in the case of Moser's entropic argument, bounds from below the entropy injected to the system while addressing the sequence. In the other part, one has to estimate the [entropy of the] set of possible witness sequences, using syntactic properties mandated by causality: roughly speaking every occurrence of a flaw in a witness sequence, with the potential exception of the very first, must be preceded by an occurrence of some flaw that causes it. Finally, one compares the rate at which the probability of a t-step witness sequence decreases (or the rate at which entropy is increased) with the rate at which the [entropy of the] set of possible witness sequences increases, to establish that their product tends to 0 with t.

In this paper, exactly because we aim to capture the intensity of interactions between flaws under adversarial noise, we need to take a different approach. In particular, our proof can be thought of as entangling the two parts described above in order to establish that, while adversarial noise can make the imposed syntactic requirements inherited by the causality graph very weak (by making the graph extremely dense), the fact that the intensity of the noise is low, suffices to control the growth rate of the entropy of the set of witness sequences. The result is a carefully tuned argument that amortizes the entropy injected into the system against its effect on the entropy of the set of Break Forests. Key to the capacity to perform this amortization is the use of so-called Break Forests, introduced in [3], which localize in time the introduction of new flaws in the state. This property of Break Forests was not used in earlier works [3, 2] and allows us to use a different amortization for the flaws introduced by the principal vs. those introduced by noise.

4 Termination via Compression

Our analysis will not depend in any way on the distribution θ of the initial state. As a result, without loss of generality, we can assume that the process starts at an arbitrary but fixed state σ_{init}. We let $A(\sigma)$ denote the support of $\rho(\sigma, \cdot)$, i.e., $A(\sigma)$ is the set of all states reachable by the process in a single step from σ.

▶ **Definition 2.** We refer to the (random) sequence $\sigma_{\text{init}} = \sigma_1, \ldots, \sigma_{t+1}$, entailing the first t steps of the process, as the *t-trajectory*. A t-trajectory is *bad* iff $\sigma_1, \ldots, \sigma_{t+1}$ are all flawed.

We model the set of all possible trajectories as an infinite tree whose root is labelled by $\sigma_1 = \sigma_{\text{init}}$. The root has $|A(\sigma_{\text{init}})|$ children corresponding to (and labelled by) each possible value of σ_2. More generally, a vertex labeled by σ has $|A(\sigma)|$ children, each child labeled by a distinct element of $A(\sigma)$, i.e., a distinct possible value of σ_{i+1}. Every edge of this infinite vertex-labelled tree is oriented away from the root and labelled by the probability of the corresponding transition, i.e., $\rho(\sigma, \tau)$, where σ is the parent and τ is the child vertex. By our assumption, every such edge label is at least 2^{-B}.

We call the above labelled infinite tree the *process tree* and note that it is nothing but the unfolding of the Markov chain corresponding to the state-evolution of the process. In particular, for every vertex v of the tree, the probability, p_v, that an infinite trajectory will go through v equals the product of the edge-labels on the root-to-v path. In visualizing the process tree it will be helpful to draw each vertex v at Euclidean distance $-\log_2 p_v$ from the root. This way all trajectories whose last vertex is at the same distance from the root are equiprobable, even though they may entail wildly different numbers of steps (this also means that sibling vertices are not necessarily equidistant from the root). Finally, we color the vertices of the process tree as follows. For every infinite path that starts at the root determine its maximal prefix forming a bad trajectory. Color the vertices of the prefix red and the remaining vertices of the path blue.

In terms of the above picture, our goal will be to prove that there exist a critical radius x_0 and $\delta > 0$, such that the proportion of red states at distance x_0 from the root is at most $1 - \delta$. Crucially, x_0 will be polynomial, in fact linear, in $m = |F|$ and $\log_2 |\Omega|$. Since we will prove this for every possible initial state and the process is Markovian, it follows that the probability that the process reaches distance x from the root while going only through red states is at most $(1 - \delta)^{\lfloor x/x_0 \rfloor}$.

To prove that red vertices thin out as we move away from the root we stratify the process tree as follows. Fix any real number $x > 0$ and on each infinite path from the root mark the first vertex of probability 2^{-x} or less, i.e., the first vertex that has distance at least x from the root. Truncate the process tree so that the marked vertices become leaves of a finite tree. Let $L(x)$ be the set of all root-to-leaf paths (trajectories) in this finite tree and let $B(x) \subseteq L(x)$ consist of the bad trajectories. Now, let I be the random variable equal to an infinite trajectory of the process and let $\Sigma = \Sigma(x)$ be the random variable equal to the prefix of I that lies in $L(x)$. By definition, $\sum_{\ell \in L(x)} \Pr[\Sigma = \ell] = 1$, while $\Pr[\ell] \in (2^{-x-B}, 2^{-x}]$ for every $\ell \in L(x)$, since $-\log_2 \rho \geq B$. Let $P = P(\Sigma)$ be the maximal red prefix of Σ and observe that if $\Sigma \in B(x)$ then $P = \Sigma$. Therefore,

$$H[P] \;\geq\; \sum_{\ell \in B(x)} \Pr[\Sigma = \ell](-\log_2 \Pr[\Sigma = \ell]) \;\geq\; x \sum_{\ell \in B(x)} \Pr[\Sigma = \ell] \;=\; x \Pr[\Sigma \in B(x)] \;. \quad (5)$$

Assume now that there exist $M_0 > 0$ and $\lambda < 1$, such that $H[P] \leq \lambda x + M_0$, for every $x > 0$. Then (5) implies that for $x_0 = 2M_0/(1 - \lambda)$,

$$\Pr[\Sigma \in B(x_0)] \leq \frac{H[P]}{x_0} \leq \frac{\lambda x_0 + M_0}{x_0} = \lambda + \frac{1 - \lambda}{2} = \frac{1 + \lambda}{2} < 1 \;. \quad (6)$$

If $\Sigma \in B(x_0)$, we treat the reached state as the root of a new finite tree and repeat the same analysis, as it is independent of the starting state. It follows in this manner that for every integer $T \geq 1$, the probability that the process reaches a state at distance $T(x_0 + B)$

or more from the root by going only through red states is at most $((1+\lambda)/2)^T$. Thus, for any $s > 1/2$, the probability that the process reaches a state at distance

$$E = \left\lceil \frac{2s}{1+\lambda} \right\rceil (x_0 + B) = O\left(\frac{sM_0}{1-\lambda^2}\right)$$

or more from the root by going only through red states is at most $((1+\lambda)/2)^{\left\lceil \frac{2s}{1+\lambda} \right\rceil} < \exp(-s)$.

Since $\rho(\sigma,\tau) < 1-2^{-B}$, it follows that $-\log_2 \rho(\sigma,\tau) > 2^{-B}$, for every arc in D. Thus, after $2^B E$ steps the process is always at distance E or more from the root. Thus, the probability of not reaching a flawless state after $2^B E = O\left(\frac{sM_0}{1-\lambda^2}\right)$ steps is $\exp(-s)$. Therefore Theorem 1 follows from the following.

▶ **Theorem 3.** *Let* $\Xi = \max\{\mathrm{b_{ns}}, \mathrm{b_{po}}\}$ *and* $\Delta = \max_{j \in F} \Delta_j$. *If there exists* $\lambda < 1$ *such that for all* $j \in [m]$,

$$\sum_{f_i \in \Gamma_{\mathrm{po}}(f_j)} 2^{-(\lambda \mathrm{Potential}(f_i) - \mathrm{b}_{\mathrm{po}}^{f_i} - q_i(p))} < 2^{-(2+h(p))} \ ,$$

then $H[P] \le \lambda x + M_0$ *for every* $x > 0$, *where* $M_0 = \log_2 |\Omega| + m(\Delta+1)(\Xi+4) + \lambda B$.

The proof of Theorem 3 can be found in the full version of the paper [1]. To bound the entropy $H[P]$ we show how to represent trajectories as break sequences, described in the next section, and then show how to bound the entropy of break sequences by showing that they can be compressed in fewer bits, on average, than those consumed by the algorithm.

5 Break Sequences

Recall that π is an arbitrary but fixed ordering of the set of flaws F and that the highest flaw present in each state σ is denoted by $\pi(\sigma)$. We will refer to $\pi(\sigma)$ as the flaw *addressed* at state σ, i.e., as in the noiseless case, even though the action distribution $P(O(\sigma))$ may be "misguided" whenever $O(\sigma) \neq U(\sigma)$.

▶ **Definition 4.** Given a bad t-trajectory Σ, its *witness* sequence is $W(\Sigma) = w_1, \ldots, w_t = \{\pi(\sigma_i)\}_{i=1}^t$.

To prove Theorem 3, i.e., to gain control of bad trajectories and thus of $H[P]$, we introduce the notion of *break sequences* (see also [3, 2]). Recall that $U(\sigma)$ denotes the set of flaws present in σ.

▶ **Definition 5.** Let $B_0 = U(\sigma_1)$. For $1 \le i \le t-1$, let $B_i = U(\sigma_{i+1}) \setminus (U(\sigma_i) \setminus w_i)$.

Thus, B_i is the set of flaws "introduced" during the i-th step, where if a flaw is addressed in a step but remains present in the resulting state we say that it "introduced itself". Each flaw $f \in B_i$ may or may not be addressed during the rest of the trajectory. For example, f may get fixed "collaterally" during some step taken to address some other flaw, before the controller had a chance to address it. Alternatively, it may be that f remains present throughout the rest of the trajectory, but in each step $i < j \le t-1$ some other flaw has greater priority than f. It will be crucial to identify and focus on the subset of flaws $B_i^* \subseteq B_i$ that *do* get addressed during the t-trajectory, causing entropy to enter the system. Per the formal Definition 6 below, the set of such flaws is $B_i^* = B_i \setminus \{O_i \cup N_i\}$, where O_i comprises any flaws in B_i that get eradicated collaterally, while N_i comprises any flaws in B_i that remain present in every subsequent state after their introduction without being addressed.

▶ **Definition 6.** The *Break Sequence* of a t-trajectory is $B_0^*, B_1^*, \ldots, B_t^*$, where for $0 \leq i \leq t$,

$$B_i^* = B_i \setminus \{O_i \cup N_i\} \text{ , where}$$
$$O_i = \{f \in B_i \mid \exists j \in [i+1, t] : f \notin U(\sigma_{j+1}) \wedge \forall \ell \in [i+1, j] : f \neq w_\ell\} \ ,$$
$$N_i = \{f \in B_i \mid \forall j \in [i+1, t] : f \in U(\sigma_{j+1}) \wedge \forall \ell \in [i+1, t] : f \neq w_\ell\} \ .$$

Given $B_0^*, B_1^*, \ldots, B_{i-1}^*$ we can determine the sequence $w_1, w_2, \ldots, w_i$ of flaws addressed inductively, as follows. Define $E_1 = B_0^*$, while for $i \geq 1$, let

$$E_{i+1} = (E_i - w_i) \cup B_i^* \ . \tag{7}$$

Observe that, by construction, $E_i \subseteq U(\sigma_i)$ and $w_i \in E_i$. Therefore, for every i, the highest flaw in E_i is w_i.

References

1 D. Achlioptas, F. Iliopoulos, and N. Vlassis. Stochastic Control via Entropy Compression. *ArXiv e-prints*, July 2016. URL: `https://arxiv.org/abs/1607.06494`, `arXiv:1607.06494`.

2 Dimitris Achlioptas and Fotis Iliopoulos. Focused stochastic local search and the Lovász local lemma. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 2024–2038. SIAM, 2016. `doi:10.1137/1.9781611974331.ch141`.

3 Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. *J. ACM*, 63(3):22, 2016. `doi:10.1145/2818352`.

4 Mikko Alava, John Ardelius, Erik Aurell, Petteri Kaski, Supriya Krishnamurthy, Pekka Orponen, and Sakari Seitz. Circumspect descent prevails in solving random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 105(40):15253–15257, 2008. `doi:10.1073/pnas.0712263105`.

5 Noga Alon. A parallel algorithmic version of the local lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991. `doi:10.1002/rsa.3240020403`.

6 József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures Algorithms*, 2(4):343–365, 1991. `doi:10.1002/rsa.3240020402`.

7 Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 2012.

8 Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.*, 234:26–48, 2016. `doi:10.1016/j.artint.2016.01.007`.

9 Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI*, pages 183–188. Citeseer, 1992.

10 Artur Czumaj and Christian Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 2000)*, pages 30–39, 2000.

11 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, Amsterdam, 1975.

12 Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

13 Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

14 David G. Harris and Aravind Srinivasan. A constructive algorithm for the Lovász local lemma on permutations. In *SODA*, pages 907–925. SIAM, 2014. doi:10.1137/1.9781611973402.68.

15 Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1327–1346. IEEE Computer Society, 2015. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7352273, doi:10.1109/FOCS.2015.85.

16 Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

17 Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

18 Kashyap Babu Rao Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *STOC*, pages 235–244. ACM, 2011. doi:10.1145/1993636.1993669.

19 Vladimir Kolmogorov. Commutativity in the algorithmic lovász local lemma. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 780–787. IEEE Computer Society, 2016. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7781469, doi:10.1109/FOCS.2016.88.

20 Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9(1):1–36, 1998.

21 Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *AAAI/IAAI*, pages 541–548, 1999.

22 Michael Molloy and Bruce Reed. Further algorithmic aspects of the local lemma. In *STOC'98 (Dallas, TX)*, pages 524–529. ACM, New York, 1999.

23 Robin A. Moser. A constructive proof of the Lovász local lemma. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 343–350. ACM, 2009. doi:10.1145/1536414.1536462.

24 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15, 2010. doi:10.1145/1667053.1667060.

25 Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM (JACM)*, 47(4):681–720, 2000.

26 Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 163–169. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185365.

27 Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

28 Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

29 Bart Selman, Henry A. Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, October 11-13, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics*

and Theoretical Computer Science, pages 521–532. DIMACS/AMS, 1993. URL: `http://dimacs.rutgers.edu/Volumes/Vol26.html`.

30 Aravind Srinivasan. Improved algorithmic versions of the Lovász local lemma. In Shang-Hua Teng, editor, *SODA*, pages 611–620. SIAM, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347150`.

31 Nikos Vlassis, Michael L. Littman, and David Barber. On the computational complexity of stochastic controller optimization in POMDPs. *ACM Transactions on Computation Theory (TOCT)*, 4(4):12, 2012. `doi:10.1145/2382559.2382563`.

Approximation Strategies for Generalized Binary Search in Weighted Trees*†

Dariusz Dereniowski[1], Adrian Kosowski[2], Przemysław Uznański[3], and Mengchuan Zou[4]

1 **Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland**
deren@eti.pg.gda.pl
2 **Inria Paris and IRIF, Université Paris Diderot, Paris, France**
adrian.kosowski@inria.fr
3 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
przemyslaw.uznanski@inf.ethz.ch
4 **Inria Paris and IRIF, Université Paris Diderot, Paris, France**
mengchuan.zou@inria.fr

Abstract

We consider the following generalization of the binary search problem. A search strategy is required to locate an unknown target node t in a given tree T. Upon querying a node v of the tree, the strategy receives as a reply an indication of the connected component of $T \setminus \{v\}$ containing the target t. The cost of querying each node is given by a known non-negative weight function, and the considered objective is to minimize the total query cost for a worst-case choice of the target. Designing an optimal strategy for a weighted tree search instance is known to be strongly NP-hard, in contrast to the unweighted variant of the problem which can be solved optimally in linear time. Here, we show that weighted tree search admits a quasi-polynomial time approximation scheme (QPTAS): for any $0 < \varepsilon < 1$, there exists a $(1+\varepsilon)$-approximation strategy with a computation time of $n^{O(\log n/\varepsilon^2)}$. Thus, the problem is not APX-hard, unless NP $\subseteq$ DTIME$(n^{O(\log n)})$. By applying a generic reduction, we obtain as a corollary that the studied problem admits a polynomial-time $O(\sqrt{\log n})$-approximation. This improves previous $\widetilde{O}(\log n)$-approximation approaches, where the $\widetilde{O}$-notation disregards $O(\text{poly}\log\log n)$-factors.

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

Keywords and phrases Approximation Algorithm, Adaptive Algorithm, Graph Search, Binary Search, Vertex Ranking, Trees

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.84

1 Introduction

In this work we consider a generalization of the fundamental problem of searching for an element in a sorted array. This problem can be seen, using graph-theoretic terms, as a problem of searching for a target node in a path, where each query reveals on which 'side' of the queried node the target node lies. The generalization we study is two-fold: a more general structure of a tree is considered and we assume non-uniform query times. Thus,

* The full version can be found at https://arxiv.org/abs/1702.08207.
† Partially supported by ANR DESCARTES and by National Science Centre (Poland) grant number 2015/17/B/ST6/01887.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 84; pp. 84:1–84:14


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

Table 1 Computational complexity of the search problem in different graph classes, including our results for weighted trees. Completeness results refer to the decision version of the problem. In the case of unweighted paths, the solution is the classical binary search algorithm.

Graph class	*Unweighted*	*Weighted*
Paths:	exact in $O(n)$ time	exact in $O(n^2)$ time [4]
		strongly NP-complete [9]
Trees:	exact in $O(n)$ time [26, 28]	$(1+\varepsilon)$-approx. in $n^{O(\log n/\varepsilon)}$ time (Thm. 3.3)
		$O(\sqrt{\log n})$-approx. in poly-time (Thm. 3.4)
Undirected:	exact in $n^{O(\log n)}$ time [10]	PSPACE-complete [10]
	$O(\log n)$-approx. in poly-time [10]	$O(\log n)$-approx. in poly-time [10]
Directed:	PSPACE-complete [10]	PSPACE-complete [10]

our problem can be stated as follows. Given a node-weighted input tree T (in which the query time of a node is provided as its weight), design a search strategy (sometimes called a decision tree) that locates a hidden *target node* x by asking *queries*. Each query selects a node v in T and after the time that equals the weight of the selected node, a reply is given: the reply is either 'yes' which implies that v is the target node and thus the search terminates, or it is 'no' in which case the search strategy receives the edge outgoing from v that belongs to the shortest path between u and v. The goal is to design a search strategy that locates the target node and minimizes the search time in the worst case.

The vertex search problem is more general than its 'edge variant' that has been more extensively studied. In the latter problem one selects an edge e of an edge-weighted tree $T = (V, E, w)$ in a query and learns in which of the two components of $T - e$ the target node is located. Indeed, this edge variant can be reduced to our problem as follows: first assign a 'large' weight to each node of T (for example, one plus the sum of the weights of all edges in the graph) and then subdivide each edge e of T giving to the new node the weight of the original edge, $w(e)$. It is apparent that an optimal search strategy for the new node-weighted tree should never query the nodes with large weights, thus immediately providing a search strategy for the edge variant of T.

We also point out that the considered problem, as well as the edge variant, being quite fundamental, were historically introduced several times under different names: minimum height elimination trees [27], ordered colourings [15], node and edge rankings [13], tree-depth [25] or LIFO-search [11]. Table 1 summarizes the complexity status of the node-query model and places our result in the general context.

1.1 State-of-the-Art

In this work we focus on the worst case search time for a given input graph and we only remark that other optimization criteria has been also considered [3, 16, 17, 30]. For other closely related models and corresponding results see e.g. [1, 12, 19, 21, 29].

The node-query model. An optimal search strategy can be computed in linear-time for an unweighted tree [26, 28]. The number of queries performed in the worst case may vary from being constant (for a star one query is enough) to being at most $\log_2 n$ for any tree [26] (by

always querying a node that halves the search space). Several following results have been obtained in [10]. First, it turns out that $\log_2 n$ queries are always sufficient for general simple graphs and this implies a $O(m^{\log_2 n} n^2 \log n)$-time optimal algorithm for arbitrary unweighted graphs. The algorithm which performs $\log_2 n$ queries also serves as a $O(\log n)$-approximation algorithm, also for the weighted version of the problem. On the other hand, it is shown in the same work that an optimal algorithm (for unweighted case) with a running time of $O(n^{o(\log n)})$ would be in contradiction with the Exponential-Time-Hypothesis. When weighted graphs are considered, the problem becomes PSPACE-complete.

The edge-query model. In the case of unweighted trees, an optimal search strategy can be computed in linear time [20, 24]. (See [7] for a correspondence between edge rankings and the searching problem.) The computational complexity of the problem on weighted trees attracted a lot of attention. On the negative side, it has been proved that it is strongly NP-hard to compute an optimal search strategy [6] for bounded diameter trees, which has been improved by showing hardness for several specific topologies: trees of diameter at most 6, trees of degree at most 3 [4] and spiders [5] (trees having at most one node of degree greater than two). On the other hand, polynomial-time algorithms exist for weighted trees of diameter at most 5 and weighted paths [4]. We note that for weighted paths there exists a linear-time but approximate solution given in [16]. For approximate polynomial-time solutions, a simple $O(\log n)$-approximation has been given in [6] and a $O(\log n/\log\log\log n)$-approximate solution is given in [4]. Then, the best known approximation ratio has been further improved to $O(\log n/\log\log n)$ in [5].

Some bounds on the number of queries for unweighted trees have been developed. Observe that an optimal search strategy needs to perform at least $\log_2 n$ queries in the worst case. However, there exist trees of maximum degree Δ that require $\Delta \log_{\Delta+1} n$ queries [2]. On the other hand, $\Theta(\Delta \log n)$ queries are always sufficient for each tree [2], which has been improved to $(\Delta+1)\log_\Delta n$ [18], $\Delta \log_\Delta n$ [8] and $1 + \frac{\Delta-1}{\log_2(\Delta+1)-1}\log_2 n$ [10].

1.2 Organization of the Paper

The aim of Section 2 is to give the necessary notation and a formal statement of the problem (Sections 2.1 and 2.2) and to provide two different but equivalent problem formulations that will be more convenient for our analysis. As opposed to the classical problem formulation in which a strategy is seen as a *decision tree*, Section 2.3 restates the problem in such a way that with each vertex v of the input tree we associate a sequence of vertices that need to be iteratively queried when v is the root of the current subtree that contains the target node. In Section 2.4 we extend this approach by associating with each vertex a sequence of not only vertices to be queried but also time points of the queries.

The latter problem formulation is suitable for a dynamic programming algorithm provided in Section 3.1. In this section we introduce an auxiliary, slightly modified measure of the cost of a search strategy. First we provide a quasi-polynomial time dynamic programming scheme that provides an arbitrarily good approximation of the output search strategy with respect to this modified cost (the analysis is deferred to Section 4), and then we prove that the new measure is sufficiently close to the original one (the analysis is deferred to Section 5). These two facts provide the quasi-polynomial time scheme for the tree search problem, achieving a $(1+\varepsilon)$-approximation with a computation time of $n^{O(\log n/\varepsilon^2)}$, for any $0 < \varepsilon < 1$.

In Section 3.2 we observe how to use the above algorithm to derive a polynomial-time $O(\sqrt{\log n})$-approximation algorithm for the tree search problem. This is done by a divide and conquer approach: a sufficiently small subtree T^* of the input tree T is first computed

so that the quasi-polynomial time algorithm runs in polynomial (in the size of T) time for T^*. This decomposes the problem: having a search strategy for T^*, the search strategies for $T - T^*$ are computed recursively.

2 Preliminaries

2.1 Notation and Query Model

We now recall the problem of searching of an unknown target node x by performing queries on the vertices of a given node-weighted rooted tree $T = (V, E, w)$ with weight function $w\colon V \to \mathbb{R}_+$. Each *query* selects one vertex v of T and after $w(v)$ time units receives an answer: either the query returns *true*, meaning that $x = v$, or it returns a neighbor u of v which lies closer to the target x than v. Since we assume that the queried graph T is a tree, such a neighbor u is unique and is equivalently described as the unique neighbor of v belonging to the same connected component of $T \setminus \{v\}$ as x.

All trees we consider are rooted. Given a tree T, the root is denoted by $r(T)$. For a node $v \in V$, we denote by T_v the subtree of T rooted at v. For any subset $V' \subseteq V$ (respectively, $E' \subseteq E$) we denote by $T[V']$ (resp., $T[E']$) the minimal subtree of T containing all nodes from V' (resp., all edges from E'). For $v \in V$, $N(v)$ is the set of neighbors of v in T.

For $U \subseteq V$ and a target node $x \notin U$, there exists a unique maximal subtree of $T \setminus U$ that contains x; we will denote this subtree by $T\langle U, x\rangle$.

We denote $|V| = n$. We will assume w.l.o.g. that the maximum weight of a vertex is normalized to 1. (This normalization is immediately obtained by a proportional scaling of all units of cost.) We will also assume w.l.o.g. that the weight function satisfies the following *star condition*: for all $v \in V$, $w(v) \leq \sum_{u \in N(v)} w(u)$. Observe that if this condition is not fulfilled, i.e., for some vertex v will have $w(v) > \sum_{u \in N(v)} w(u)$, then vertex v will never be queried by any optimal strategy in v, since a query to v can then be replaced by a sequence of queries to all neighbors of v, obtaining not less information at strictly smaller cost. In general, given an instance which does not satisfy the star condition, we enforce it by performing all necessary weight replacements $w(v) \leftarrow \min\{w(v), \sum_{u \in N(v)} w(u)\}$, for $v \in V$. Replacements terminate definitely since no vertex will be replaced more than once.

For $a, \omega \in \mathbb{R}_{\geq 0}$, we denote the rounding of a down (up) to the nearest multiple of ω as $\lfloor a \rfloor_\omega = \omega \lfloor a/\omega \rfloor$ and $\lceil a \rceil_\omega = \omega \lceil a/\omega \rceil$, respectively.

2.2 Definition of a Search Strategy

A search strategy $\mathcal{A}$ for a rooted tree $T = (V, E, w)$ is an adaptive algorithm which defines successive queries to the tree, based on responses to previous queries, with the objective of locating the target vertex in a finite number of steps. Note that search strategies can be seen as decision trees in which each node represents a subset of vertices of T that contains x, with leaves representing singletons consisting of x.

Let $\mathtt{Q}_\mathcal{A}(T, x)$ be the time-ordering (sequence) of queries performed by strategy $\mathcal{A}$ on tree T to find a target vertex x, with $\mathtt{Q}_{\mathcal{A},i}(T, x)$ denoting the i-th queried vertex in this time ordering, $1 \leq i \leq |\mathtt{Q}_\mathcal{A}(T, x)|$.

We denote by $\mathtt{COST}_\mathcal{A}(T, x) = \sum_{i=1}^{|\mathtt{Q}_\mathcal{A}(T,x)|} w(\mathtt{Q}_{\mathcal{A},i}(T, x))$ the sum of weights of all vertices queried by $\mathcal{A}$ with x being the target node, i.e., the time after which $\mathcal{A}$ finishes. Let $\mathtt{COST}_\mathcal{A}(T) = \max_{x \in V} \mathtt{COST}_\mathcal{A}(T, x)$ be the *cost of* $\mathcal{A}$. We define the *cost of* T to be $\mathtt{OPT}(T) = \min\{\mathtt{COST}_\mathcal{A}(T) \mid \mathcal{A} \text{ is a search strategy for } T\}$. We say that a search strategy is *optimal* for T if its cost equals $\mathtt{OPT}(T)$.

Algorithm 2.1 Search strategy $\mathcal{A}_S$ for a query sequence assignment S.

```
1: v ← r(T)    // stores current root
2: U ← ∅
3: while |T⟨U, x⟩| > 1 do
4:     for u ∈ S(v) do
5:         if u ∈ T⟨U, x⟩ then    // u is the first vertex in S(v) that belongs to T⟨U, x⟩
6:             QueryVertex(u)
7:             U ← U ∪ {u}
8:             if v ≠ r(T⟨U, x⟩) then    // query reply is 'down'
9:                 v ← r(T⟨U, x⟩)
10:                break    // for loop
```

As a consequence of normalization and the star condition, we have the following bound.

▶ **Observation 2.1.** *For any tree T, we have* $1 \leq \mathtt{OPT}(T) \leq \lceil \log_2 n \rceil$.

All omitted proofs are provided in the full version.

We also introduce the following notation. If the first $|U|$ queried vertices by a search strategy $\mathcal{A}$ are exactly the vertices in U, $U = \{\mathtt{Q}_{\mathcal{A},i}(T, x) : 1 \leq i \leq |U|\}$, then we say that $\mathcal{A}$ *reaches* $T\langle U, x\rangle$ *through* U, and $w(U)$ is the *cost of reaching* $T\langle U, x\rangle$ *by* $\mathcal{A}$. We also say that we receive an 'up' reply to a query to a vertex v if the root of the tree remaining to be searched remains unchanged by the query, i.e., $r(T\langle U, x\rangle) = r(T\langle U \cup \{v\}, x\rangle)$, and we call the reply a 'down' reply when the root of the remaining tree changes, i.e., $r(T\langle U, x\rangle) \neq r(T\langle U \cup \{v\}, x\rangle)$.

2.3 Query Sequences and Stable Strategies

By a slight abuse of notation, we will call a search strategy *polynomial-time* if it can be implemented using a dynamic (adaptive) algorithm which computes the next queried vertex in polynomial time.

We give most of our attention herein to search strategies in trees which admit a natural (non-adaptive, polynomial-space) representation called a *query sequence assignment*. Formally, for a rooted tree T, the *query sequence assignment* S is a function $S : V \to V^*$, which assigns to each vertex $v \in V$ an ordered sequence of vertices $S(v)$, known as the *query sequence* of v. The query sequence assignment directly induces a strategy $\mathcal{A}_S$, presented as Algorithm 2.1. Intuitively, the strategy processes successive queries from the sequence $S(v)$, where v is the root vertex of the current search tree, $v = r(T\langle U, x\rangle)$, where U is the set of queries performed so far. This processing is performed in such a way that the strategy iteratively takes the first vertex in $S(v)$ that belongs to $T\langle U, x\rangle$ and queries it. As soon as the root of the search tree changes, the procedure starts processing queries from the sequence of the new root, which belong to the remaining search tree. The procedure terminates as soon as $T\langle U, x\rangle$ has been reduced to a single vertex, which is necessarily the target x.

In what follows, in order to show that our approximation strategies are polynomial-time, we will confine ourselves to presenting a polynomial-time algorithm which outputs an appropriate sequence assignment.

A sequence assignment is called *stable* if the replacement of line 9 in Algorithm 2.1 by any assignment of the form $v \leftarrow v''$, where v'' is an arbitrary vertex which is promised to lie on the path from $r(T\langle U, x\rangle)$ to the target x, always results in a strategy which performs a (not necessarily strict) subsequence of the sequence of queries performed by the original strategy $\mathcal{A}_S$. Sequence assignments computed on trees with a bottom-up approach usually have the stability property; we provide a proof of stability for one of our main routines in Section 4.

Without loss of generality, we will also assume that if $v \in S(v)$, then v is the last element of $S(v)$. Indeed, when considering a subtree rooted at v, after a query to v, if v was not the target, then the root of the considered subtree will change to one of the children of v, hence any subsequent elements of $S(v)$ may be removed without changing the strategy.

2.4 Strategies Based on Consistent Schedules

Intuitively, we may represent search strategies by a schedule consisting of some number of jobs, with each job being associated to querying a node in the tree (cf. e.g. [14, 22, 23]). Each job has a fixed processing time, which is set to the weight of a node. Formally, in this work we will refer to the schedule $\hat{S}$ only in the very precise context of search strategies $\mathcal{A}_S$ based on some query sequence assignment S. The *schedule assignment* $\hat{S}$ is the following extension of the sequence assignment S, which additionally encodes the starting time of any search query job. If the query sequence S of a node v is of the form $S(v) = (v_1, \ldots, v_k)$, $k = |S(v)|$, then the corresponding schedule for v will be given as $\hat{S}(v) = ((v_1, t_1), \ldots, (v_k, t_k))$, with $t_i \in \mathbb{R}_{\geq 0}$ denoting the starting time of the query for v_i. We will call $\hat{S}(v)$ the *schedule of node* v. We will call a schedule assignment $\hat{S}$ *consistent* with respect to search in a given tree T if the following conditions are fulfilled:

- **(i)** No two jobs in the schedule of a node overlap: for all $v \in V$, for two distinct jobs $(u_1, t_1), (u_2, t_2) \in \hat{S}(v)$, we have $|[t_1, t_1 + w(u_1)] \cap [t_2, t_2 + w(u_2)]| = 0$.
- **(ii)** If v is the parent of v' in T and $(u, t) \in \hat{S}(v')$, then we either also have $(u, t) \in \hat{S}(v)$, or the job $(v, t_v) \in \hat{S}(v)$ completes before the start of job (u, t): $t_v + w(v) \leq t$.

It follows directly from the definition that a consistent schedule assignment (and the underlying query sequence assignment) is uniquely determined by the collection of jobs $\{(v, t_v) : (v, t_v) \in \hat{S}(u), u \in V\}$. Note that not every vertex has to contain a query to itself in its schedule; we will occasionally write $t_v = \perp$ to denote that such a job is missing.

By extension of notation for sequence assignments, we will denote a strategy following a consistent schedule assignment $\hat{S}$ (i.e., executing the query jobs of schedule $\hat{S}$ at the prescribed times) as $\mathcal{A}_{\hat{S}}$. We will then have: $\texttt{COST}_{\mathcal{A}_{\hat{S}}}(T) = |\hat{S}|$, where $|\hat{S}|$ is the *duration* of schedule assignment $\hat{S}$, given as: $|\hat{S}| = \max_{v \in V} |\hat{S}(v)|$, with: $|\hat{S}(v)| = \max_{(u,t) \in \hat{S}(v)} (t + w(u))$.

We remark that there always exists an optimal search strategy which is based on a consistent schedule. By a well-known characterization (cf. e.g. [6]), tree T satisfies $\texttt{OPT}(T) = \tau \in \mathbb{R}$ if and only if there exists an assignment $I : V \to \mathcal{I}_\tau$ of intervals of time to nodes before deadline τ, $\mathcal{I}_\tau = \{[a, b] : 0 \leq a < b \leq \tau\}$, such that $|I(v)| = w(v)$ and if $|I(u) \cap I(v)| > 0$ for any pair of nodes $u, v \in V$, then the $u - v$ path in T contains a separating vertex z such that $\max I(z) \leq \min(I(u) \cup I(v))$. The corresponding schedule assignment of duration τ is obtained by adding, for each node $u \in V$, the job $(u, \min I(u))$ to the schedule of all nodes on the path from u towards the root, until a node v such that $\max I(v) \leq \min I(u)$ is encountered on this path. The consistency and correctness of the obtained schedule is immediate to verify.

▶ **Observation 2.2.** *For any tree T, there exists a query sequence assignment S and a corresponding consistent schedule $\hat{S}$ on T such that $|\hat{S}| = \texttt{OPT}(T)$.*

3 The Results

3.1 $(1 + \varepsilon)$-Approximation in $n^{O(\log n/\varepsilon^2)}$ Time

We first present an approximation scheme for the weighted tree search problem with $n^{O(\log n)}$ running time. The main difficulty consists in obtaining a constant approximation ratio for

the problem with this running time; we at once present this approximation scheme with tuned parameters, so as to achieve $(1+\varepsilon)$-approximation in $n^{O(\log n/\varepsilon^2)}$ time.

Our construction consists of two main building blocks. First, we design an algorithm based on a bottom-up (dynamic programming) approach, which considers exhaustively feasible sequence assignments and query schedules over a carefully restricted state space of size $n^{O(\log n)}$ for each node. The output of the algorithm provides us both with a lower bound on $\texttt{OPT}(T)$, and with a sequence assignment-based strategy $\mathcal{A}_S$ for solving the tree search problem. The performance of this strategy $\mathcal{A}_S$ is closely linked to the performance of $\texttt{OPT}(T)$, however, there is one type of query, namely a query on a vertex of small weight leading to a 'down' response, due to whose repeated occurrence the eventual cost difference between $\texttt{COST}_{\mathcal{A}_S}(T)$ and $\texttt{OPT}(T)$ may eventually become arbitrarily large. To alleviate this difficulty, we introduce an alternative measure of cost which compensates for the appearance of the disadvantageous type of query.

We start by introducing some additional notation. Let $\omega \in \mathbb{R}_+$, be an arbitrarily fixed value of weight and let $c \in \mathbb{N}$. The choice of constant $c \in \mathbb{N}$ will correspond to an approximation ratio of $(1+\varepsilon)$ of the designed scheme for $\varepsilon = 168/c$.

We say that a query to a vertex v is a *light down query* in some strategy if $w(v) < c\omega$ and $x \in V(T_v)$, i.e., it is also a 'down' query, where x is the target vertex.

For any strategy $\mathcal{A}$, we denote by $\texttt{COST}^{(\omega,c)}_{\mathcal{A}}(T,x)$ its modified cost of finding target x, defined as follows. Let d_x be the number of light down queries when searching for x: $d_x = \left|\{i : w(\mathtt{Q}_{\mathcal{A},i}(T,x)) < c\omega \text{ and } x \in V(T_{\mathtt{Q}_{\mathcal{A},i}(T,x)})\}\right|$. Then, the modified cost $\texttt{COST}^{(\omega,c)}_{\mathcal{A}}(T,x)$ is:

$$\texttt{COST}^{(\omega,c)}_{\mathcal{A}}(T,x) = \texttt{COST}_{\mathcal{A}}(T,x) - (2c+1)\omega d_x. \tag{1}$$

and by a natural extension of notation: $\texttt{COST}^{(\omega,c)}_{\mathcal{A}}(T) = \max_{x\in V} \texttt{COST}^{(\omega,c)}_{\mathcal{A}}(T,x)$.

The technical result which we will obtain in Section 4 may now be stated as follows.

▶ **Proposition 3.1.** *For any $c \in \mathbb{N}$, $L \in \mathbb{N}$, there exists an algorithm running in time $(cn)^{O(L)}$, which for any tree T constructs a stable sequence assignment S and computes a value of ω such that $\omega \le \frac{1}{L}\texttt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T)$ and: $\texttt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) \le \left(1+\frac{12}{c}\right)\texttt{OPT}(T)$.*

In order to convert the obtained strategy $\mathcal{A}_S$ with a small value of $\texttt{COST}^{(\omega,c)}$ into a strategy with small $\texttt{COST}$, we describe in Section 5 an appropriate strategy conversion mechanism. The approach we adopt is applicable to any strategy based on a stable sequence assignment and consists in concatenating, for each vertex $v \in V$, a prefix to the query sequence $S(v)$ in the form of a separately computed sequence $R(v)$, which does not depend on $S(v)$. The considered query sequences are thus of the form $R(v) \circ S(v)$, where the symbol "$\circ$" represents sequence concatenation. Intuitively, the sequences R, taken over the whole tree, reflect the structure of a specific solution to the unweighted tree search problem on a contraction of tree T, in which each edge connecting a node to a child with weight at least $c\omega$ is contracted. We recall that the optimal number of queries to reach a target in an unweighted tree is $O(\log n)$, and the goal of this conversion is to reduce the number of light down queries in the combined strategy to at most $O(\log n)$.

▶ **Proposition 3.2.** *For any fixed $\omega > 0$ there exists a polynomial-time algorithm which for a tree T computes a sequence assignment $R : V \to V^*$, such that, for any strategy $\mathcal{A}_S$ based on a stable sequence assignment S, the sequence assignment S^+, given by $S^+(v) = R(v) \circ S(v)$ for each $v \in V$, has the following property:*

$$\texttt{COST}_{\mathcal{A}_{S^+}}(T) \le \texttt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) + 4(2c+1)\omega \log_2 n.$$

The proof of Proposition 3.2 is provided in Section 5.

We are now ready to put together the two bounds. Combining the claims of Proposition 3.1 for $L = \lceil c^2 \log_2 n \rceil$ (with $\omega \leq \frac{1}{L}\mathtt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) \leq \frac{\mathtt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T)}{c^2 \log_2 n}$) and Proposition 3.2, we obtain:

$$\begin{aligned}\mathtt{COST}_{\mathcal{A}_{S^+}}(T) &\leq \mathtt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) + 4(2c+1)\omega \log_2 n \leq \mathtt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) + 12c\omega \log_2 n \;\leq \\ &\leq \mathtt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) + 12c \log_2 n \frac{\mathtt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T)}{c^2 \log_2 n} \leq \left(1 + \frac{12}{c}\right) \mathtt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) \leq \\ &\leq \left(1 + \frac{12}{c}\right)^2 \mathtt{OPT}(T) \leq \left(1 + \frac{168}{c}\right) \mathtt{OPT}(T).\end{aligned}$$

After putting $\varepsilon = \frac{168}{c}$ and noting that in stating our result we can safely assume $c = O(\mathrm{poly}(n))$ (beyond this, the tree search problem can be trivially solved optimally in $O(n^n)$ time using exhaustive search), we obtain the main theorem of the section.

▶ **Theorem 3.3.** *There exists an algorithm running in $n^{O(\frac{\log n}{\varepsilon^2})}$ time, providing a $(1+\varepsilon)$-approximation solution to the weighted tree search problem for any $0 < \varepsilon < 1$.*

3.2 Extension: A Poly-Time $O(\sqrt{\log n})$-Approximation Algorithm

We now present the second main result of this work. By recursively applying the previously designed QPTAS (Theorem 3.3) with $\varepsilon = 1$, we obtain a polynomial-time $O(\sqrt{\log n})$-approximation algorithm for finding search strategy for an arbitrary weighted tree. We start by informally sketching the algorithm – we follow here the general outline of the idea from [5]. The algorithm is recursive and starts by finding a minimal subtree T^* of an input tree whose removal disconnects T into subtrees, each of size bounded by $n/2^{\sqrt{\log n}}$. The tree T^* will be processed by our QPTAS algorithm described in Section 3.1. This results either in locating the target node, if it belongs to T^*, or identifying the component of $T - T^*$ containing the target, in which case the search continues recursively in the component. Subtrees considered at each level of recursion are disjoint, thus factors of approximation add up over recursion levels. However, for the final algorithm to have polynomial running time, the tree T^* needs to be of size $2^{O(\sqrt{\log n})}$. This is obtained by contracting paths in T^* (each vertex of the path has at most two neighbors in T^*) into single nodes having appropriately chosen weights. Since T^* has $2^{O(\sqrt{\log n})}$ leaves, this narrows down the size of T^* to the required level and we argue that an optimal search strategy for the 'contracted' T^* provides a search strategy for the original T^* that is within a constant factor from the cost of T^*.

A formal exposition and analysis of the obtained algorithm is provided in the full version.

▶ **Theorem 3.4.** *There is a $O(\sqrt{\log n})$-approximation polynomial time algorithm for the weighted tree search problem.*

4 Quasi-Polynomial Computation of Strategies with Small $\mathtt{COST}^{(\omega,c)}$

4.1 Preprocessing: Time Alignment in Schedules

We adopt here a method similar but arguably more refined than rounding techniques in scheduling problems of combinatorial optimization, showing that we could discretize the starting and finishing time of jobs, as well as weights of vertices, in a way to restrict the size of state space for each node to $n^{O(\log n)}$, without introducing much error.

Fix $c \in \mathbb{N}$ and $\omega = \frac{a}{cn}$ for some $a \in \mathbb{N}$. (In subsequent considerations, we will have $c = \Theta(1/\varepsilon)$, $a = O(\frac{n}{\log n})$ and $\omega = \Omega(\varepsilon/\log n)$.) Given a tree $T = (V, E, w)$, let $T' = (V, E, w')$ be a tree with the same topology as T but with weights rounded up as follows:

$$w'(v) = \begin{cases} \lceil w(v) \rceil_{\omega}, & \text{if } w(v) > c\omega, \\ \lceil w(v) \rceil_{\frac{1}{cn}}, & \text{otherwise.} \end{cases} \tag{2}$$

We will informally refer to vertices with $w(v) > c\omega$ (equivalently $w'(v) > c\omega$) as *heavy vertices* and vertices with $w(v) \leq c\omega$ (equivalently $w'(v) \leq c\omega$) as *light vertices*. (Note that $w(v) \leq c\omega$ if and only if $w'(v) \leq c\omega$.)

When designing schedules, we consider time divided into *boxes* of duration ω, with the i-th box equal to $[i\omega, (i+1)\omega]$. Each box is divided into a identical *slots* of length $\frac{1}{cn}$.

In the tree T', the duration of a query to a heavy vertex is an integer number of boxes, and the duration of a query to a light vertex is an integer number of slots. We next show that, without affecting significantly the approximation ratio of the strategy, we can align each query to a heavy vertex in the schedule so that it occupies an interval of full adjacent boxes, and each query to a light vertex in the schedule so that it occupies an interval of full adjacent slots (possibly contained in more than one box).

We start by showing the relationship between the costs of optimal solutions for trees T and T'.

▶ **Lemma 4.1.** $\texttt{OPT}(T) \leq \texttt{OPT}(T') \leq (1 + \frac{2}{c})\texttt{OPT}(T)$.

▶ **Lemma 4.2.** *There exists a consistent schedule assignment $\hat{S}$ for tree T' such that $\texttt{COST}_{\mathcal{A}_{\hat{S}}}(T') \leq (1 + \frac{3}{c})\texttt{OPT}(T')$ and for all $v \in V$ we have that*

- *if $w'(v) > c\omega$, (v is heavy), then the starting time t of any job (v, t) in the schedule $\hat{S}(u)$ of any $u \in V$ is an integer multiple of ω (aligned to a box),*
- *if $w'(v) \leq c\omega$, (v is light), then the starting time t of any query (v, t) in the schedule $\hat{S}(u)$ of any $u \in V$ is an integer multiple of $\frac{1}{cn}$ (aligned to a slot).*

A schedule on tree T' satisfying the conditions of Lemma 4.2, and the resulting search strategy, are called *aligned*. Subsequently, we will design an aligned strategy on tree T', and compare the quality of the obtained solution to the best aligned strategy for T'.

The intuition between the separate treatment of heavy vertices (aligned to boxes) and light vertices (aligned to slots) in aligned schedules is the following. Whereas the time ordering of boxes is essential in the design of the correct strategy, in our dynamic programming approach we will not be concerned about the order of slots within a single box (i.e., the order of queries to light vertices placed in a single box). This allows us to reduce the state space of a node. Whereas the ordering of slots in the box will eventually have to be repaired to provide a correct strategy, this will not affect the quality of the overall solution too much (except for the issue of light down queries pointed out earlier, which are handled separately in Section 5).

4.2 Dynamic Programming Routine for Fixed Box Size

Let the values of parameter c and box size ω be fixed as before. Additionally, let $L \in \mathbb{N}$ be a parameter representing the time limit for the duration of the considered vertex schedules when measured in boxes, i.e., the longest schedule considered by the procedure will be of length $L\omega$ (we will eventually choose an appropriate value of $L = O(\log n)$ as required when showing Theorem 3.3).

In order to lower-bound the duration of the consistent aligned schedule assignment with minimum cost, we perform an exhaustive bottom-up evaluation of aligned schedules which

satisfy constraints on the occupancy of slots. However, instead of considering individual slots of a schedule which may be empty or full, for reasons of efficiency we consider the *load* $s_v[p]$ of each box, $0 \leq p < L$, in the same schedule, defined informally as the proportion of the duration of the occupied slots within the box to the duration ω of the box. In the full version, we formally show the following claim.

▶ **Lemma 4.3.** *Assume that the data structure $(s_v, t_v)_{v \in V}$ corresponds to a consistent schedule. Let $v \in V$ be an arbitrarily chosen node with set of children $\{v_1, \ldots, v_l\}$. Then the set of queried nodes forms an edge cover of the tree:*

$$\text{If } t_v = \perp, \text{ then } t_{v_j} \neq \perp, \text{ for all } 1 \leq j \leq l. \tag{3}$$

Moreover, let completion time t^v_{end} of the query to v given as:

$$t^v_{end} = \begin{cases} t_v + w'(v), & \text{if } t_v \neq \perp, \\ +\infty, & \text{if } t_v = \perp. \end{cases}$$

Let a_p be the contribution to the load of the p-th time box of the query job for vertex v, i.e.

$$a_p = \begin{cases} \frac{1}{\omega} |[t_v, t^v_{end}] \cap [p\omega, (p+1)\omega]| & \text{if } t_v \neq \perp, \\ 0 & \text{if } t_v = \perp. \end{cases}$$

Then, for any box $[p\omega, (p+1)\omega]$, $0 \leq p < L$, we have the following bounds on the amount of load which can be packed into the box:

$$\left.\begin{array}{ll} s_v[p] = a_p + \sum_{j=1}^{l} s_{v_j}[p] \in [0,1], & \text{when } t^v_{end} \geq (p+1)\omega, \\ s_v[p] \geq a_p, & \text{when } p\omega < t^v_{end} < (p+1)\omega, \\ s_v[p] = 0, & \text{when } t^v_{end} \leq p\omega. \end{array}\right\} \tag{4}$$

Moreover, for any box $[p\omega, (p+1)\omega]$, $0 \leq p < L$, we have that the total load of a query to v and queries propagated from any of the subtrees cannot exceed 1:

$$\text{For all } 1 \leq j \leq l, \text{ the following bound holds: } s_{v_j}[p] + a_p \leq 1. \tag{5}$$

We now show that the shortest schedule assignments satisfying the set of constraints (3), (4), and (5) can be found in $n^{O(\log n)}$ time. This is achieved by using the procedure BuildStrategy, presented in Algorithm 4.1, which returns for a node v a non-empty set of schedules $\hat{\mathcal{S}}[v]$, such that each $s_v \in \hat{\mathcal{S}}[v]$ can be extended into the sought assignment of schedules in its subtree, $(s_u, t_u)_{u \in V(T_v)}$. In the statement of Algorithm 4.1, we recall that, given a tree $T = (V, E, w)$, tree $T' = (V, E, w')$ is the tree with weights rounded up to the nearest multiple of the length of a slot (see Equation (2)).

The subsequent steps taken in procedure BuildStrategy can be informally sketched as follows. The input tree T' is processed in a bottom-up manner and hence, for an input vertex v, the recursive calls for its children $v_1, \ldots, v_l$ are first made, providing schedule assignments for the children (see lines 3–4). Then, the rest of the pseudocode is responsible for using these schedule assignments to obtain all valid schedule assignments for v. Lines 10–14 merge the schedules of the children in such a way that a set $\hat{S}^*_i$, $i \in \{1, \ldots, l\}$, contains all schedule assignments computed on the basis of the schedules for the children $v_1, \ldots, v_i$. Thus, the set $\hat{S}^*_l$ is the final product of this part of the procedure and is used in the remaining part. Note

Algorithm 4.1 Dynamic programming routine BUILDSTRATEGY for a tree T'. $L, c \in \mathbb{N}$ are global parameters. Subroutines MERGESCHEDULES and INSERTVERTEX are provided in the full version.

```
1: procedure BuildStrategy(vertex v, box size ω ∈ ℝ)
2:     l ← number of children of v in T'    // Denote by v_1, ..., v_l the children of v.
3:     for i = 1..l do
4:         Ŝ[v_i] ← BuildStrategy(v_i, ω);
5:     s ← 0^L
6:     s.max_child_load ← 0^L
7:     s.must_contain_v ← false
8:     Ŝ_0 ← {s}    // Ŝ_0 contains the schedule with no queries.
9:     // Inductively, Ŝ_i^* is based on merging schedules at v_1, ..., v_i.
10:    for i = 1..l do
11:        Ŝ_i^* ← ∅
12:        for each schedule s ∈ Ŝ_{i-1}^* do
13:            for each schedule s_add ∈ Ŝ[v_i] do
14:                Ŝ_i^* ← Ŝ_i^* ∪ MergeSchedules(s, s_add, ω);
15:    Ŝ[v] ← ∅
16:    for each s ∈ Ŝ_l^* do
17:        if w'(v) > cω then    // v is heavy
18:            for p = 0..L − 1 do    //attempt to insert (into s) query to v starting from time-box p
19:                Ŝ[v] ← Ŝ[v] ∪ InsertVertex(s, v, ω, p · ω)
20:        else    //v is light
21:            for real t = 0..L · ω step 1/(cn) do
22:                //attempt to insert (into s) query to v at a slot from time t
23:                Ŝ[v] ← Ŝ[v] ∪ InsertVertex(s, v, ω, t)
24:        if s.must_contain_v = false then
25:            Ŝ[v] ← Ŝ[v] ∪ InsertVertex(s, v, ω, ⊥)
26:    return Ŝ[v]
```

that a schedule assignment in $\hat{S}_l^*$ may not be valid since a query to v is not accommodated in it – the rest of the pseudocode is responsible for taking each schedule $s \in \hat{S}_l^*$ and inserting a query to v into s. More precisely, the subroutine INSERTVERTEX is used to place the query to v at all possible time points (depending whether v is heavy or light). We note that the subroutine MERGESCHEDULES, for each schedule s it produces, sets a Boolean 'flag' $s.must\_contain\_v$ that whenever equals $false$, indicates that querying v is not necessary in s to obtain a valid schedule for v (this happens if s queries all children of v). A detailed analysis of procedure BUILDSTRATEGY can be found in the full version.

▶ **Lemma 4.4.** *For fixed constants $L, c \in \mathbb{N}$, calling procedure* BUILDSTRATEGY$(r(T), \omega)$*, where $r(T)$ is the root of the tree, determines if there exists a tuple $(s_v, t_v)_{v \in V}$ which satisfies constraints* (3)*,* (4)*, and* (5)*, or returns an empty set otherwise.*

It follows directly from Lemma 4.4 that, for any value ω^*, tree T may only admit an aligned schedule assignment of duration at most $\omega^* L$ if a call to procedure BUILDSTRATEGY $(r(T), \omega^*)$ returns a non-empty set. Taking into account Lemmas 4.1 and 4.2, we directly obtain the following lower bound on the length of the shortest aligned schedule in tree T'.

▶ **Lemma 4.5.** *If* BUILDSTRATEGY$(r(T), \omega^*) = \emptyset$*, then:*

$$\omega^* L < \left(1 + \frac{3}{c}\right) \texttt{OPT}(T') \leq \left(1 + \frac{3}{c}\right)\left(1 + \frac{2}{c}\right) \texttt{OPT}(T) \leq \left(1 + \frac{11}{c}\right) \texttt{OPT}(T).$$

▶ **Lemma 4.6.** *The running time of procedure* BuildStrategy$(r(T), \omega)$ *is at most* $O((cn)^{\gamma L})$*, for some absolute constant* $\gamma = O(1)$*, for any* $\omega \leq n$*.*

To complete the proof of Proposition 3.1, we can now provide a strategy which achieves a small value of $\texttt{COST}^{(\omega,c)}$. This relies on procedure BuildStrategy$(r(T), \omega)$ as an essential subroutine, first determining the minimum value of $\omega = \frac{i}{cn}$, $i \in \mathbb{N}$, for which BuildStrategy produces a schedule. Details of the approach are provided in the full version.

5 Reducing the Number of Down-Queries

We start with defining a function $\ell\colon V \to \{1, \ldots, \lceil \log_2 n \rceil\}$ which in the following will be called a *labeling of* T and the value $\ell(v)$ is called the *label of* v. We say that a subset of nodes $H \subseteq V$ is an *extended heavy part* in T if $H = \{v\} \cup H'$, where all nodes in H' are heavy, no node in H' has a heavy neighbor in T that does not belong to H' and v is the parent of some node in H'. Let $H_1, \ldots, H_l$ be all extended heavy parts in T. Obtain a tree $T_C = (V_C, E_C)$ by contracting, in T, the subgraph H_i into a node denoted by h_i for each $i \in \{1, \ldots, l\}$. In the tree T_C, we want to find its labeling $\ell'\colon V_C \to \{1, \ldots, \lceil \log_2 |V_C| \rceil\}$ that satisfies the following condition: for each two nodes u and v in V_C with $\ell'(u) = \ell'(v)$, the path between u and v has a node z satisfying $\ell'(z) < \ell'(u)$. One can obtain such a labeling by a following procedure that takes a subtree T'_C of T_C and an integer i as an input. Find a central node v in T'_C, set $\ell'(v) = i$ and call the procedure for each subtree T''_C of $T'_C - v$ with input T''_C and $i + 1$. The procedure is initially called for input T and $i = 1$. We also remark that, alternatively, such a labeling can be obtained via vertex rankings [13, 28].

Once the labeling ℓ' of T_C is constructed, we extend it to a labeling ℓ of T in such a way that for each node v of T we set $\ell(v) = \ell'(v)$ if $v \notin H_1 \cup \cdots \cup H_l$ and $\ell(v) = \ell'(h_i)$ if $v \in H_i$, $i \in \{1, \ldots, l\}$.

Having the labeling ℓ of T, we are ready to define a query sequence $R(v)$ for each node $v \in V$. $R(v)$ contains all nodes u from T_v such that $\ell(u) < \ell(v)$ and each internal node z of the path connecting v and u in T satisfies $\ell(z) > \ell(u)$. Additionally, the nodes in $R(v)$ are ordered by increasing values of their labels.

By x we refer to the target node in T. Fix S to be a stable sequence assignment in the remaining part of this section and by R we refer to the sequence assignment constructed above. Then, we fix S^+ to be $S^+(v) = R(v) \circ S(v)$ for each $v \in V$. A query made by $\mathcal{A}_{S^+}$ to a node that belongs to $R(v)$ for some $v \in V$ is called an *R-query*; otherwise it is an *S-query*. In the full version we show that, in $\mathcal{A}_{S^+}$, the total number of R-queries does not exceed $2\log_2 n$. Moreover, since S is stable, for each target node x, the S-queries performed by $\mathcal{A}_{S^+}$ are a subsequence of the queries performed by $\mathcal{A}_S$. Therefore, the potentially additional queries made by $\mathcal{A}_{S^+}$ with respect to $\mathcal{A}_S$ are R-queries. We then formally show that each R-query is made on a light node and that any R-query increases the value of $\texttt{COST}^{(\omega,c)}$ of $\mathcal{A}_{S^+}$ with respect to the value of $\texttt{COST}^{(\omega,c)}$ of $\mathcal{A}_S$ by at most $(2c+1)\omega$. Hence we have: $\texttt{COST}^{(\omega,c)}_{\mathcal{A}_{S^+}}(T) \leq \texttt{COST}^{(\omega,c)}_{\mathcal{A}_S}(T) + 2(2c+1)\omega \log_2 n$.

Moreover, we show in the full version that the total number of queries in strategy $\mathcal{A}_{S^+}$ to light nodes receiving 'down' replies can be likewise bounded by $2\log_2 n$. Since each such query introduces a rounding difference of at most $(2c+1)\omega$ when comparing cost functions $\texttt{COST}$ and $\texttt{COST}^{(\omega,c)}$, we thus obtain: $\texttt{COST}_{\mathcal{A}_{S^+}}(T) \leq \texttt{COST}^{(\omega,c)}_{\mathcal{A}_{S^+}}(T) + 2(2c+1)\omega \log_2 n$.

Combining the above observations gives the claim of the Proposition.

References

1 Esther M. Arkin, Henk Meijer, Joseph S. B. Mitchell, David Rappaport, and Steven Skiena. Decision trees for geometric models. *Int. J. Comput. Geometry Appl.*, 8(3):343–364, 1998. doi:10.1142/S0218195998000175.

2 Yosi Ben-Asher and Eitan Farchi. The cost of searching in general trees versus complete binary trees. Technical report, Technical report, 1997.

3 Ferdinando Cicalese, Tobias Jacobs, Eduardo Sany Laber, and Marco Molinaro. On the complexity of searching in trees and partially ordered structures. *Theor. Comput. Sci.*, 412(50):6879–6896, 2011. doi:10.1016/j.tcs.2011.08.042.

4 Ferdinando Cicalese, Tobias Jacobs, Eduardo Sany Laber, and Caio Dias Valentim. The binary identification problem for weighted trees. *Theor. Comput. Sci.*, 459:100–112, 2012. doi:10.1016/j.tcs.2012.06.023.

5 Ferdinando Cicalese, Balázs Keszegh, Bernard Lidický, Dömötör Pálvölgyi, and Tomás Valla. On the tree search problem with non-uniform costs. *Theor. Comput. Sci.*, 647:22–32, 2016. doi:10.1016/j.tcs.2016.07.019.

6 Dariusz Dereniowski. Edge ranking of weighted trees. *Discrete Applied Mathematics*, 154(8):1198–1209, 2006. doi:10.1016/j.dam.2005.11.005.

7 Dariusz Dereniowski. Edge ranking and searching in partial orders. *Discrete Applied Mathematics*, 156(13):2493–2500, 2008. doi:10.1016/j.dam.2008.03.007.

8 Dariusz Dereniowski and Marek Kubale. Efficient parallel query processing by graph ranking. *Fundam. Inform.*, 69(3):273–285, 2006.

9 Dariusz Dereniowski and Adam Nadolski. Vertex rankings of chordal graphs and weighted trees. *Inf. Process. Lett.*, 98(3):96–100, 2006. doi:10.1016/j.ipl.2005.12.006.

10 Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 519–532, 2016. doi:10.1145/2897518.2897656.

11 Archontia C. Giannopoulou, Paul Hunter, and Dimitrios M. Thilikos. Lifo-search: A min-max theorem and a searching game for cycle-rank and tree-depth. *Discrete Applied Mathematics*, 160(15):2089–2097, 2012. doi:10.1016/j.dam.2012.03.015.

12 Brent Heeringa, Marius Catalin Iordan, and Louis Theran. Searching in dynamic tree-like partial orders. In *Algorithms and Data Structures – 12th International Symposium, WADS 2011, New York, NY, USA, August 15-17, 2011. Proceedings*, pages 512–523, 2011. doi:10.1007/978-3-642-22300-6_43.

13 Ananth V. Iyer, H. Donald Ratliff, and Gopalakrishnan Vijayan. Optimal node ranking of trees. *Inf. Process. Lett.*, 28(5):225–229, 1988. doi:10.1016/0020-0190(88)90194-9.

14 Ananth V. Iyer, H. Donald Ratliff, and Gopalakrishnan Vijayan. Parallel assembly of modular products – an analysis. Technical report, Technical Report 88-86, Georgia Institute of Technology, 1988.

15 Meir Katchalski, William McCuaig, and Suzanne M. Seager. Ordered colourings. *Discrete Mathematics*, 142(1-3):141–154, 1995. doi:10.1016/0012-365X(93)E0216-Q.

16 Eduardo Sany Laber, Ruy Luiz Milidiú, and Artur Alves Pessoa. On binary searching with nonuniform costs. *SIAM J. Comput.*, 31(4):1022–1047, 2002. doi:10.1137/S0097539700381991.

17 Eduardo Sany Laber and Marco Molinaro. An approximation algorithm for binary searching in trees. *Algorithmica*, 59(4):601–620, 2011. doi:10.1007/s00453-009-9325-0.

18 Eduardo Sany Laber and Loana Tito Nogueira. Fast searching in trees. *Electronic Notes in Discrete Mathematics*, 7:90–93, 2001. doi:10.1016/S1571-0653(04)00232-X.

19 Eduardo Sany Laber and Loana Tito Nogueira. On the hardness of the minimum height decision tree problem. *Discrete Applied Mathematics*, 144(1-2):209–212, 2004. doi:10.1016/j.dam.2004.06.002.

20 Tak Wah Lam and Fung Ling Yue. Optimal edge ranking of trees in linear time. *Algorithmica*, 30(1):12–33, 2001. doi:10.1007/s004530010076.

21 Nathan Linial and Michael E. Saks. Searching ordered structures. *J. Algorithms*, 6(1):86–103, 1985. doi:10.1016/0196-6774(85)90020-3.

22 Joseph W. H. Liu. Computational models and task scheduling for parallel sparse cholesky factorization. *Parallel Computing*, 3(4):327–342, 1986. doi:10.1016/0167-8191(86)90014-1.

23 Joseph W. H. Liu. The role of elimination trees in sparse factorization. *SIAM. J. Matrix Anal. & Appl.*, 11(1):134–172, 1990.

24 Shay Mozes, Krzysztof Onak, and Oren Weimann. Finding an optimal tree searching strategy in linear time. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 1096–1105, 2008.

25 Jaroslav Nesetril and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006. doi:10.1016/j.ejc.2005.01.010.

26 Krzysztof Onak and Pawel Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 379–388, 2006. doi:10.1109/FOCS.2006.32.

27 Alex Pothen. The complexity of optimal elimination trees. Technical report, Technical Report CS-88-13, Pennsylvannia State University, 1988.

28 Alejandro A. Schäffer. Optimal node ranking of trees in linear time. *Inf. Process. Lett.*, 33(2):91–96, 1989. doi:10.1016/0020-0190(89)90161-0.

29 George Steiner. Searching in 2-dimensional partial orders. *J. Algorithms*, 8(1):95–105, 1987. doi:10.1016/0196-6774(87)90029-0.

30 Jayme Luiz Szwarcfiter, Gonzalo Navarro, Ricardo A. Baeza-Yates, Joísa de S. Oliveira, Walter Cunto, and Nivio Ziviani. Optimal binary search trees with costs depending on the access paths. *Theor. Comput. Sci.*, 290(3):1799–1814, 2003. doi:10.1016/S0304-3975(02)00084-1.

Tighter Hard Instances for PPSZ

Pavel Pudlák[*1], **Dominik Scheder**[†2], **and Navid Talebanfard**[‡3]

1 **Czech Academy of Sciences, Prague, Czech Republic**
pudlak@math.cas.cz
2 **Shanghai Jiaotong University, Shanghai, China**
dominik@cs.sjtu.edu.cn
3 **Czech Academy of Sciences, Prague, Czech Republic**
talebanfard@math.cas.cz

Abstract

We construct uniquely satisfiable k-CNF formulas that are hard for the PPSZ algorithm, the currently best known algorithm solving k-SAT. This algorithm tries to generate a satisfying assignment by picking a random variable at a time and attempting to derive its value using some inference heuristic and otherwise assigning a random value. The "weak PPSZ" checks all subformulas of a given size to derive a value and the "strong PPSZ" runs resolution with width bounded by some given function. Firstly, we construct graph-instances on which "weak PPSZ" has savings of at most $(2+\epsilon)/k$; the *saving* of an algorithm on an input formula with n variables is the largest γ such that the algorithm succeeds (i.e. finds a satisfying assignment) with probability at least $2^{-(1-\gamma)n}$. Since PPSZ (both weak and strong) is known to have savings of at least $\frac{\pi^2+o(1)}{6k}$, this is optimal up to the constant factor. In particular, for $k=3$, our upper bound is $2^{0.333\ldots n}$, which is fairly close to the lower bound $2^{0.386\ldots n}$ of Hertli [SIAM J. Comput.'14]. We also construct instances based on linear systems over $\mathbb{F}_2$ for which *strong* PPSZ has savings of at most $O\left(\frac{\log(k)}{k}\right)$. This is only a $\log(k)$ factor away from the optimal bound. Our constructions improve previous savings upper bound of $O\left(\frac{\log^2(k)}{k}\right)$ due to Chen et al. [SODA'13].

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases k-SAT, Strong Exponential Time Hypothesis, PPSZ, Resolution

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.85

1 Introduction

The k-SAT problem is one of the most fundamental NP-complete problems: given a k-CNF formula decide if there is an assignment to the variables that satisfies all the clauses. While a simple exhaustive search algorithm solves the problem, attempting to beat this trivial approach remains an active direction (see e.g. [11, 12, 4, 10]). Formalizing the true hardness of k-SAT, Impagliazzo and Paturi [7] presented two hypotheses: *Exponential Time Hypothesis (ETH)* which rules out any $2^{o(n)}$ time algorithm for k-SAT where n is the number of variables and *Strong Exponential Time Hypothesis (Strong ETH)* which says that for any $\epsilon > 0$ there exists $k > 0$ such that k-SAT cannot be solved in time $2^{(1-\epsilon)n}$. Both ETH and Strong ETH

* The author is supported by the grant P202/12/G061 of GAČR and by the institute grant RVO:67985840.
† Dominik Scheder gratefully acknowledges support by the National Natural Science Foundation of China under grant 61502300.
‡ Supported by supported by the institute grant RVO:67985840. Part of the work was done while Navid Talebanfard was with Tokyo Institute of Technology and visiting Saint Petersburg State University during the special semester in complexity.

have successfully been used to explain the hardness of many other problems; under ETH one can prove tight lower bounds for many fixed parameter tractable problems (see [8]), and under Strong ETH several lower bounds for polynomial time solvable problems are proved (see e.g. [1]). However the validity of both these hypotheses remains a matter of mystery and in particular regarding Strong ETH no consensus seems to be within reach any time soon.

In this paper we focus on Strong ETH and the problem of constructing hard instances for known classes of algorithms for k-SAT. Paturi, Pudlák, Saks and Zane [10] presented the currently best known randomized algorithm for k-SAT. The algorithm roughly does the following: pick uniformly at random a variable x from the input formula. Try to infer the value of x using some sound heuristic. If this check fails, pick a random value for x. Set x to be this value and repeat. A sound heuristic is an algorithm P that receives a formula F and a variable x such that $P(F,x)=0$ implies $F \models (x=0)$ and $P(F,x)=1$ implies $F \models (x=1)$. We will consider two heuristics, P_w^{weak} amd P_w^{strong}, where P_p^{weak} checks if the value of x can be derived from any set of w clauses of F, and P_w^{strong} checks if the value of x can be derived by a width-w resolution derivation from F. Note that if F is $O(1)$-CNF then both P_w^{weak} and P_w^{strong} run in subexponential time as long as $w = o(\frac{n}{\log n})$. The first result showing that even simple sound heuristics can yield non-trivial savings over exhaustive search was proved by Paturi, Pudlák and Zane.

▶ **Theorem 1** ([11]). *Let F be a k-CNF formula on n variable. Then*

$$\Pr[\texttt{ppsz}(F, P_1^{\text{weak}}) \in \text{sat}(F)] \geq 2^{-(1-\frac{1}{k})n}.$$

Naturally one can ask if stronger heuristics can improve the success probability. It was indeed shown in the following theorem that using $\omega(1)$-width resolution yields improvements.

▶ **Theorem 2** ([10]). *Let F be a k-CNF formula on n variables. Then*

$$\Pr[\texttt{ppsz}(F, P_{\omega(1)}^{\text{strong}}) \in \text{sat}(F)] \geq 2^{-(1-\frac{\pi^2}{6k}-o(1))n}.$$

Later Hertli [6] showed among other things that even $P_{\omega(1)}^{\text{weak}}$ yields the same improvement over the trivial P_1^{weak}.

▶ **Theorem 3** ([6]). *Let F be a k-CNF formula on n variables. Then*

$$\Pr[\texttt{ppsz}(F, P_{\omega(1)}^{\text{weak}}) \in \text{sat}(F)] \geq 2^{-(1-\frac{\pi^2}{6k}-o(1))n}.$$

The first construction of hard instances for PPSZ was given by Chen, Scheder, Talebanfard and Tang [3]. These instances are hard even for $P_{\omega(1)}^{\text{strong}}$.

▶ **Theorem 4** ([3]). *For any large enough $k, n > 0$ there are k-CNF formulas F such that*

$$\Pr[\texttt{ppsz}(F, P_{n/k}^{\text{strong}}) \in \text{sat}(F)] \leq 2^{-(1-O(\log^2 k/k))n}.$$

In this paper we improve this upper bound. For $P_{\omega(1)}^{\text{weak}}$ we give completely different constructions for which we can show that the success probability of PPSZ is essentially tight. For $P_{\omega(1)}^{\text{strong}}$ we can improve the asymptotics of k from $O(\log^2 k/k)$ to $O(\log k/k)$.

▶ **Theorem 5.** *For every $k \geq 3$ and every large enough n there exists a uniquely satisfiable k-CNF formula F on n variables such that*

1. $\Pr[\texttt{ppsz}(F, P_w^{\text{weak}}) \in \text{sat}(F)] \leq 2^{-(1-\frac{2}{k})n}$ *for some* $w = \Theta(\log n)$,
2. *for any* $\epsilon > 0$, $\Pr[\texttt{ppsz}(F, P_w^{\text{weak}}) \in \text{sat}(F)] \leq 2^{-(1-\frac{2(1+\epsilon)}{k})n}$ *for some* $w = n^{\Theta(\epsilon)}$.

In particular, for $k = 3$, our upper bound is $2^{0.333\ldots n}$, which is fairly close to the lower bound $2^{0.386\ldots n}$ of [6].

▶ **Theorem 6.** *For every $k \geq 3$ and every large enough n, there exists a k-CNF formula F on n variables with a unique satisfying assignment such that $\texttt{ppsz}(F, P^{\text{strong}}_{n/k})$ is successful with probability at most $2^{(-1+\epsilon)n}$, where $\epsilon = O\left(\frac{\log(k)}{k}\right)$.*

Unfortunately we fail to obtain $2^{-(1-O(1/k))n}$ upper bounds for strong PPSZ. Is this possible at all or indeed strong PPSZ succeeds with probability at least $2^{-(1-\omega(1/k))n}$?

The analysis of our hard instances is based on an encoding view of PPSZ. Given a formula F on variables $x_1, \ldots, x_n$ and a satisfying assignment $\mathbf{b}$, PPSZ produces an encoding of the assignment with respect to a given permutation π of the variables in the following way.

```
encode(b, π, F, P)
c := empty string
for i = 1, ..., n do
    if P(F, x_π(i)) ∉ {0, 1} then
        append b_π(i) to c;
    end
    F := F|_{x_π(i) → b_π(i)};
end
```

It is not hard to see that we can express the success probability of PPSZ in terms of expected code lengths as follows.

▶ **Lemma 7** ([10]). *Let F be a k-CNF and let P be a sound heuristic. We have* $\Pr[\texttt{ppsz}(F, P) \in \text{sat}(F)] = \sum_{\mathbf{b} \in \text{sat}(F)} \mathbb{E}_\pi \, 2^{-|\texttt{encode}(\mathbf{b}, \pi, F, P)|}$.

Thus our goal is to construct instances having a few satisfying assignments, all of which admitting only long encodings. Defining the optimal encoding length a satisfying assignment $\mathbf{b}$ to be $\text{codelength}(F, P, \mathbf{b}) := \min_\pi |\texttt{encode}(\mathbf{b}, \pi, F, P)|$ we get

$$\Pr[\texttt{ppsz}(F, P) \in \text{sat}(F)] \quad \leq \quad \sum_{\mathbf{b} \in \text{sat}(F)} 2^{-\text{codelength}(\mathbf{b}, F, P)} \, .$$

The formulas in Theorem 5 and Theorem 6 have the unique satisfying assignment $\mathbf{0}$. Thus our goal will be to prove a lower bound on $\text{codelength}(F, P, \mathbf{0})$.

2 Notation and Preliminaries

Let F be a CNF formula with variable set V. A restriction (or partial assignment) is a partial function $\rho : V \to \{0, 1\}$. For $\mathbf{b} \in \{0, 1\}^n$, the notation $S \mapsto \mathbf{b}$ is the restriction that maps $x \in S$ to $\mathbf{b}_x$ and is undefined on $V \setminus S$. By $F|_\rho$ we denote the formula arising from fixing the variables according to ρ and then simplifying the resulting formula by removing unsatisfied literals and satisfied clauses. For a matrix $A \in \mathbb{F}_2^{m \times n}$ and $U \subseteq [n]$ we denote by A_U the $(m \times |U|)$ submatrix formed by taking all columns indexed by some $i \in U$. By $\mathbf{0}$ we denote the all-0-assignment as well as the null vector in $\mathbb{F}_2^n$.

We will identify a vector $\mathbf{a} \in \mathbb{F}_2^n$ with its support $\{i \in [n] \mid a_i = 1\}$. Thus we will liberally write things like $\mathbf{a} \cup \mathbf{b}$, $\mathbf{a} \setminus \mathbf{b}$, $|\mathbf{a}|$, and so on.

We list some key observations relating PPSZ and resolution. The (easy) proofs can be found in the appendix.

▶ **Definition 8.** Let F be a formula with a unique satisfying assignment, which without loss of generality is $\mathbf{0}$, and let P be a proof heuristic. We say F *collapses under* P if there is an ordering $x_1, \ldots, x_n$ of the variables in F such that $F|_{(x_1,\ldots,x_{i-1}\mapsto 0)} \vdash_P (x_i = 0)$ for all $1 \le i \le n$.

▶ **Proposition 9.** *If* $\mathrm{codelength}(F, P, \mathbf{b}) \le m$ *then there is a set* S *of* m *variables such that* $F|_{S\mapsto\mathbf{b}}$ *collapses under* P.

The next lemma states that if F collapses "sequentially" under bounded-width resolution, then it collapses "simultaneously" as well.

▶ **Proposition 10.** *Let* F *be a* k*-CNF formula with the unique satisfying assignment* $\mathbf{0}$*, and let* $w \ge k$*. If* F *collapses under* P_w^{strong} *then* $F \vdash_w^{\mathrm{strong}} (x = 0)$ *for all variables* x *of* F.

The next proposition connects logical implication and collapse under P^{weak} to linear algebra.

▶ **Proposition 11.** *Let* $A \in \mathbb{F}_2^{m\times n}$ *and* F_A *be its linear formula. If* $\vdash_w^{\mathrm{weak}} (F, x_i) \in \{0, 1\}$ *then there is a row vector* $\mathbf{r} \in \mathbb{F}_2^m$ *of Hamming weight at most* w *such that* $\mathbf{r} \cdot A = \mathbf{e}_i$.

3 Hard Instances for Weak PPSZ: Proof of Theorem 5

The construction in this section is based on a modification of satisfiable Tseitin formulas. Unsatisfiable Tseitin formulas are extensively studied in proof complexity (see e.g. [13, 14]). Given a graph $G = (V, E)$, the girth of G is defined as the size of the shortest cycle in G. We denote this by $g(G)$. For every pair $e, e' \in E(G)$ of edges we define the distance between e and e' by $\min_{u\in e, v\in e'}\{d(u, v)\}$. We will need graphs of bounded degree with large girth. According to a well-known result of Erdős and Sachs [5], for every $k \ge 3$ and every sufficiently large n, there exists a k-regular graph with n vertices and girth $> \log_{k-1} n$. Explicit constructions for infinitely many values of k with a better constant are also known [9].

Given a degree-k graph $G = (V, E)$, the Tseitin formula $T(G)$ is defined as follows. For each edge $e \in E$, there is a propositional variable x_e. For each vertex $v \in V$ we add the constraint $\sum_{e\ni v} x_e = 0 \pmod 2$, which can be written as a conjunction of 2^{k-1} k-clauses. [1] In our formulas we assume that the girth of the graph is at least $\log_{k-1} n$, where n denotes the number of vertices. Furthermore, we add a clause $\neg x_e \vee \neg x_{e'}$ for each pair of edges e, e' of distance at least $\frac{g(G)}{2} - 1$ (which is $\ge \frac{1}{2}\log_{k-1} n - 1$). We call these clauses *bridges* and we denote the conjunction of all of them by B. Define $F_G := T(G) \wedge B$. Note that F_G has $N = kn/2$ variables.

The following proposition follows readily.

▶ **Proposition 12.** F_G *has the unique satisfying assignment* $\mathbf{0}$.

Proof. For an assignment α let G_α denote the spanning subgraph of G containing the edges e with $\alpha(e) = 1$. Note that α satisfies $T(G)$ if and only if G_α is *even*, i.e., every vertex has even degree. There are two cases: either G_α is the empty graph, in which case $\alpha = \mathbf{0}$ and satisfies F_G, too. Or G_α contains a cycle C, which has length at least $g(G)$ and therefore contains a bridge. In this case, α violates B. ◀

[1] The original Tseitin tautologies express the fact that the system $\sum_{e\ni v} x_e = a_v \pmod 2$ is unsatisfiable if $\sum_v a_v = 1 \pmod 2$.

We will consider PPSZ with P_w^{weak} when $w = O(\log n)$

▶ **Lemma 13.** *In F_G any encoding of the all-0 assignment has length at least $(1 - \frac{2}{k})N$ under $w \leq \frac{1}{2}\log_{k-1} n - 1$.*

Proof. Suppose for contradiction that $\text{codelength}(F, P_w^{\text{weak}}) \leq \left(1 - \frac{2}{k}\right) N$ collapses. Then there is a restriction ρ that sets some $\left(1 - \frac{2}{k}\right) N$ variables to 0 such that $F_G|_\rho$ collapses under P_w^{weak}. Note that ρ leaves at least $\frac{2}{k}N = n$ variables (i.e., edge) unset. Obviously this set of edges contains a cycle C. Let $\sigma := (E \setminus C \mapsto \mathbf{0})$. Clearly $F_G|_\sigma$ also collapses. This contradicts the next lemma:

▶ **Lemma 14.** *Let $C \subseteq E$ be a cycle and $\sigma := (E \setminus C \mapsto \mathbf{0})$. Then $F_G|_\sigma$ does not collapse under P_w^{weak}.*

Proof. Suppose it does collapse. Then there exists an edge $e \in C$ and a set $F' \subseteq F|_\sigma$ of at most w clauses such that $F' \models \neg x_e$.

The clauses in F' are either coming from the Tseitin part or from the bridges. Consider a path $P = v_1, \ldots, v_s$ of maximum length on which e appears and the vertices of P are mentioned by Tseitin clauses in F'. Note that P cannot contain the whole cycle, since otherwise there would be too many clauses in F'. Let v_0 and v_{s+1} be vertices on $C \setminus P$ connected to v_1 and v_s, respectively. We extend P by v_0 and v_{s+1}. Since $w \leq \frac{1}{2}\log_{k-1} n - 1$, there is no bridge between any pair of edges appearing on P. We can now simply set all the variables in P to 1 and all other variables to 0. This would satisfy F' and yet it sets x_e to 1, contradicting that $F' \models \neg x_e$. ◀

This concludes the proof of Lemma 13. ◀

Below we show that it is possible to obtain similar lower bounds even when w is some function in $n^{O(\epsilon)}$.

▶ **Lemma 15.** *For every $\epsilon > 0$ and every sufficiently large n, any encoding of the all-0 assignment with $w < n^{\frac{\epsilon}{8(k-1)}}$ has length at least $(1 - \frac{2(1+\epsilon)}{k})N$.*

Proof. Let S be the set of edges appearing in any encoding of the all-0 assignment. We will show that $|E \setminus S| < (1+\epsilon)n$. Assume for a contradiction that $|E \setminus S| \geq (1+\epsilon)n$. We will show that $E \setminus S$ contains a large subgraph which is expanding in a certain sense.

▶ **Definition 16.** In a graph G we say that a path $P = v_1, \ldots, v_t$ is *slender* if for all $1 \leq i \leq t$ we have $d(v_i) \leq 2$.

▶ **Lemma 17.** *Let $G = (V, E)$ be a graph on n vertices such that $|E| \geq (1+\epsilon)n$ for some $\epsilon > 0$. There exists an induced subgraph $H \subseteq G$ on at least $\Omega(\epsilon^{3/4} n^{1/4})$ vertices with $\delta(G) \geq 2$ with no slender path of length $\geq 2/\epsilon$.*

Proof. Let $r = 2/\epsilon$. We first find a subgraph of minimum degree at least 2 on at least $\Omega(\sqrt{n/r})$ vertices with many edges. To do this we can remove vertices of degree at most 1 at a time. Having removed t vertices we are left with a graph on $n - t$ vertices and at least $(1 + \frac{2}{r})n - t$ edges. It holds that $(1 + \frac{2}{r})n - t \geq (1 + \frac{2}{r})(n - t)$. As the remaining graph has at most $\binom{n-t}{2}$ edges we have $(1 + 2/r)n \leq \binom{n-t}{2} + t$. This implies $t \leq n - \Omega(\sqrt{n/r})$. Let $n' = n - t$. We thus have $n' \geq \Omega(\sqrt{n/r})$.

If the remaining graph has no slender path of length r we are done. Otherwise let $v_1, \ldots, v_{t_1}$ be a maximal slender path, i.e., $d(v_i) = 2$ for all $1 \leq i \leq t_1$ and v_1 and v_{t_1} have a neighbor (possibly the same) outside P of degree at least 3. We remove $v_1, \ldots, v_{t_1}$ from the

graph. If there are any vertices of degree 1 we remove them one at a time until there are no more such vertices. Let the total number removed vertices be t'_1. We repeat this for d rounds until there are no more slender paths of length r and all vertices have degree at least 2. Let t_i and t'_i be defined similarly for the ith iteration. We have $t'_i \geq r$ and thus $d \leq n'/r$. Note that the total number of removed edges is $t'_1 + \ldots + t'_d + d$ and hence at most $n' + n'/r$. We are left with a graph with at least n'/r edges and hence at least $\Omega(\sqrt{n'/r}) = \Omega(\epsilon^{3/4} n^{1/4})$ vertices. ◀

Applying Lemma 17 on $E \setminus S$ we obtain a subgraph H with minimum degree at least 2 which does not contain any slender path of length $\geq 2/\epsilon$. Setting all edges outside of H to 0, we obtain that there exists a set of at most w clauses F' in the restricted formula which implies $x_e = 0$ for some $e \in H$. Let $e = (u, v)$. We will construct a tree T_{uv} in H by growing two disjoint rooted trees T_u and T_v, starting at u and v, respectively. The crucial requirement is that in both T_u and T_v any path of length $\geq 2/\epsilon$ that goes downwards in the rooted tree there exists a vertex of degree ≥ 3. We call such a vertex a *branching* vertex. Furthermore, in T_{uv} the distance between the first branching vertices in T_u and T_v is at most $2/\epsilon$. Using the fact that the minimum degree in H is at least 2 and it does not contain any slender path of length $2/\epsilon$ and that the girth is at least $\log_{k-1} n$, we can easily construct T_{uv} so that each root to leaf path in both T_u and T_v has $\frac{\epsilon}{8} \log_{k-1} n$ branching vertices. Since the horizon $w < n^{\frac{\epsilon}{8(k-1)}}$, there are vertices u' and v' in T_u and T_v, respectively, that are not mentioned in F'. Consider the unique path between u' and v' in T_{uv}. Note that this path has length at most $\frac{1}{2} \log_{k-1} n$. However, since we put bridges only between edges of distance more that $\frac{1}{2} \log_{k-1} n$, there is no bridge between any pair of edges on this path. Setting all edges on the path including e to 1 and everything else to 0 satisfies F', contradicting to $F' \models \neg x_e$. ◀

Lemma 13 implies that $\text{codelength}(F_G, P_w^{\text{weak}}) \geq (1 - \frac{2}{k})N$ for $w \leq \frac{1}{2} \log_{k-1} n - 1$. Similarly, Lemma 15 implies that $\text{codelength}(F_G, P_w^{\text{weak}}) \geq (1 - \frac{2(1+\epsilon)}{k})N$ for $w < n^{\frac{\epsilon}{8(k-1)}}$. This completes the proof of Theorem 5.

4 Hard Linear Formulas for Strong PPSZ: Proof of Theorem 6

Suppose $A \in \mathbf{F}^{m \times n}$ is a matrix in which every row has Hamming weight at most k. Then the system $A \cdot \mathbf{x} = \mathbf{0}$ consists of m linear equations over n variables, each of which involves at most k variables. One can encode it as a k-CNF formula with $2^{k-1} \cdot m$ clauses. Let us denote this formula by F_A. A CNF formula which in this way encodes a system of linear equations will be called a *linear CNF* formula.

4.1 Robust Expanding Matrices

As often in the realm of resolution, our proof of hardness relies on a certain notion of expansion. Loosely speaking, a matrix A is a *robust expander* if for every "sufficiently large" submatrix A_U and every "sufficiently diverse" set of row vectors $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ at least one of the vectors $\mathbf{u}_i \cdot A_U$ has "large" Hamming weight. We will now define this notion formally. Throughout this section, let $k \in \mathbb{N}$ be arbitrary but fixed (this is the k for which we want to construct hard k-CNF formulas). A sequence $\mathbf{u}_1, \ldots, \mathbf{u}_\ell \in \mathbb{F}_2^n$ is *well-increasing* if $n/k \leq |\mathbf{u}_i \setminus (\mathbf{u}_1 \cup \cdots \cup \mathbf{u}_{i-1})| \leq 4n/k$ for every $1 \leq i \leq \ell$. This is what we mean by "sufficiently diverse".

▶ **Definition 18** (Robust Expanders). A matrix $A \in \mathbb{F}_2^{n\times n}$ is called a t-robust (ℓ, w)-expander if for every $U \subseteq [n]$ of size t and well-increasing sequence $\mathbf{u}_1 \ldots, \mathbf{u}_\ell$, there is some $1 \leq i \leq \ell$ such that $|\mathbf{u}_i \cdot A_U| > w$.

▶ **Theorem 19** (Robust Expanders Are Hard). *Let $t, w \in \mathbb{N}$, $w \geq 2n/k$, and $\ell := \lfloor \frac{k\cdot t}{4n} \rfloor$. If A is a t-robust (ℓ, w)-expander, then* $\mathrm{codelength}(F, P_w^{\mathrm{strong}}, \mathbf{0}) \geq n - t$.

▶ **Theorem 20** (Robust Expanders Exist). *For every sufficiently large n, there is a matrix $A \in \mathbb{F}_2^{n\times n}$ such that (1) every row of A has Hamming weight at most $k+1$; (2) the rank of A is at least $n - 2\log(n)$; (3) A is a t-robust (ℓ, w)-expander for $t = \frac{60\cdot\log(k)}{k} \cdot n$, $\ell = \lfloor \frac{k\cdot t}{4n} \rfloor$, and $w = 2n/k$.*

With these theorems we can prove Theorem 6 for strong PPSZ. Write $t = \frac{60\cdot\log(k)}{k} \cdot n$ and let A be a matrix as promised by Theorem 20. By Theorem 19 we know that $\mathrm{codelength}(F_A, P_{2n/k}^{\mathrm{strong}}, \mathbf{0}) \geq n - t$. The Steinitz exchange lemma from linear algebra gives us $2\log(n)$ unit row vectors that we can add to A to obtain a matrix $A' \in F_2^{(n+2\log(n))\times n}$ of row rank n. This means that $F_{A'}$ has the unique satisfying assignment $\mathbf{0}$. Each added unit row vector in A' is a unit clause in $F_{A'}$. It can easily be verified that adding a unit clause reduces codelength by at most 1. Therefore $\mathrm{codelength}(F_{A'}, P_{n/k}^{\mathrm{weak}}, \mathbf{0}) \geq \mathrm{codelength}(F_A, P_{n/k}^{\mathrm{weak}}, \mathbf{0}) - 2\log(n) \geq n - t - 2\log(n)$. This proves Theorem 6.

Proof of Theorem 19. Let P be the strong proof heuristic which performs resolution of width up to w. We assume that $\mathrm{codelength}(F_A, P, \mathbf{0}) \leq n - t$ and will derive a contradiction to the assumption that A is a robust expander.

By Proposition 9 and 10, $\mathrm{codelength}(F_A, P, \mathbf{0}) \leq n - t$ means that there is a partition $[n] = U \uplus S$ with $|U| = t$ such that $F' \vdash_P (x_i = 0)$ for every $i \in U$, where $F' := F_A|_{S\mapsto\mathbf{0}}$ is the formula obtained from F by setting every variable in S to 0. For notational simplicity assume $U = \{1, \ldots, t\}$. By a connection between resolution and linear algebra which is folklore by now (see e.g. [2]), the fact that $F' \vdash_P (x_i = 0)$ means the following:

▶ **Proposition 21** (Connection Between Resolution and Linear Algebra). *For every $i \in U$ there exists a binary tree T_i in which every node v is labeled with a row vector $\mathbf{r}_v \in \mathbb{F}_2^n$ such that:*

1. *for a leaf v, the label $\mathbf{r}_v$ is a unit vector,*
2. *if v is an inner node and v_0, v_1 are its children then $\mathbf{r}_v = \mathbf{r}_{v_0} + \mathbf{r}_{v_1}$.*
3. *$|\mathbf{r}_v \cdot A_U| \leq w$ for every node v of T_i,*
4. *$\mathbf{r}_{\mathrm{root}} \cdot A_U = \mathbf{e}_i$.*

We call T_i the resolution tree of x_i.

For $i \in \{1, \ldots, t\}$ let $\mathbf{r}_i$ be the root labels of the tree T_i. Since $\mathbf{r}_i \cdot A_U = \mathbf{e}_i$ we conclude that the vectors $\mathbf{r}_1, \ldots, \mathbf{r}_t$ are linearly independent. In particular this means that $|\mathbf{r}_1 \cup \cdots \cup \mathbf{r}_t| \geq t$. Equipped with these observations and the previous proposition, we can now construct a well-increasing sequence $\mathbf{u}_1, \ldots, \mathbf{u}_{\ell^*}$ with $\ell^* := \lfloor \frac{k\cdot t}{4n} \rfloor$ and $|\mathbf{u}_i \cdot A_U| \leq w$ for all $1 \leq i \leq \ell^*$. This will be a contradiction to the assumption that A is a robust expander.

Start with the empty sequence and $\ell = 0$. While $\ell < \ell^*$, we try to extend the current well-increasing sequence $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ by considering two cases. For convenience let $\mathbf{u} = \mathbf{u}_1 \cup \cdots \cup \mathbf{u}_\ell$. Note that $\frac{\ell\cdot n}{k} \leq |\mathbf{u}| \leq \frac{4\ell\cdot n}{k}$.

Case 1. Suppose some vector $\mathbf{r}_i$ among $\mathbf{r}_1, \ldots, \mathbf{r}_t$ satisfies $|\mathbf{r}_i \setminus \mathbf{u}| > 2n/k$. Recall that $\mathbf{r}_i$ is the root label of the tree T_i. We walk from the root of T_i to a leaf by always choosing the child v for which the "weight" $|\mathbf{r}_v \setminus \mathbf{u}|$ is largest. Note that this weight is more than $2n/k$ at

the root and at most 1 at a leaf. Also, in every step the weight decreases by at most a factor of 2. Thus we find a node v on the path for which $n/k \leq |\mathbf{r}_v \setminus \mathbf{u}| \leq 2n/k$. We set $\mathbf{u}_{\ell+1} := \mathbf{r}_v$ and see that the sequence $\mathbf{u}_1, \dots, \mathbf{u}_{\ell+1}$ is well-increasing by the choice of $\mathbf{u}_{\ell+1}$. Also, since $\mathbf{u}_{\ell+1}$ is the label of a node in a resolution tree, it holds that $|\mathbf{u}_{\ell+1} \cdot A_U| \leq w$.

Case 2. Suppose $|\mathbf{r}_i \setminus \mathbf{u}| \leq 2n/k$ for all $1 \leq i \leq t$. Since $|(\mathbf{r}_1 \cup \dots \cup \mathbf{r}_t) \setminus \mathbf{u}| \geq t - |\mathbf{u}| \geq t - \frac{4\ell \cdot n}{k} \geq t - \frac{4\ell^* \cdot n}{k} + \frac{4n}{k} \geq \frac{4n}{k}$, we can find a subset $I \subseteq [t]$ with $|I| \leq 2n/k$ such that $2n/k \leq \left|\bigcup_{i \in I} \mathbf{r}_i \setminus \mathbf{u}\right| \leq 4n/k$. If we let v be a random linear combination of the $\mathbf{r}_i, i \in I$, we see that $\mathbb{E}[|\mathbf{v} \setminus \mathbf{u}|] \geq n/k$. Thus, there is some vector $\mathbf{v}$ which is a linear combination of the $\mathbf{r}_i, i \in I$ and $n/k \leq |\mathbf{v} \setminus \mathbf{u}| \leq 4n/k$. Furthermore, since $|\mathbf{r}_i \cdot A_U| = |\mathbf{e}_i| = 1$ we get $|\mathbf{v} \cdot A_U| \leq \sum_{i \in I} |\mathbf{r}_i \cdot A_U| = |I| \leq 2n/k \leq w$. We can extend the sequence $\mathbf{u}_1, \dots, \mathbf{u}_\ell$ by setting $\mathbf{u}_{\ell+1} = \mathbf{v}$.

To summarize, this iteratively constructs a well-increasing sequence $\mathbf{u}_1, \dots, \mathbf{u}_{\ell^*}$ with $|\mathbf{u}_i \cdot A_U| \leq w$. We obtain a contradiction to the assumption that A is a robust expander, which completes the proof. ◀

4.2 Robust Kernel Expanders Exist – Proof of Theorem 20

Proof of Theorem 20. We will show that a matrix A sampled from a suitable probability distribution is a t-robust (ℓ, w) expander with high probability, for $t = \frac{60 \cdot \log(k)}{k} \cdot n$, $\ell = 5\log(k)$, and $w = 2n/k$. Note that by definition, this will also be a t-robust (ℓ', w)-expander for every $\ell' \geq \ell$, thus also for $\ell' = \left\lfloor \frac{k \cdot t}{4n} \right\rfloor = \left\lfloor \frac{60}{4} \cdot \log(k) \right\rfloor \geq 5\log(k) = \ell$.

Take a step k random walk in the Hamming cube $\{0,1\}^n$ and let $\mathbf{X}$ be its endpoint. We view $\mathbf{X}$ as a row vector in $\mathbb{F}_2^n$. Repeating this experiment n times independently gives n row vectors that form a matrix $B \in \mathbb{F}_2^{n \times n}$. Surely each row of B has Hamming weight at most k, and B turns out to be a robust expander. Unfortunately its kernel will have dimension $\Theta\left(\frac{\log^2(k) n}{k}\right)$ on expectation—too large for our purposes. We introduce a nice trick that boosts the rank of B.

▶ **Lemma 22.** *Let $B \in \mathbb{F}_2^{n \times n}$ be a matrix and let P be a random permutation matrix. Then $\mathbb{E}[|\ker(B+P)|] \leq n+1$.*

Proof. The kernel of a matrix $A \in \mathbb{F}_2^{n \times n}$ is the set $\{\mathbf{x} \in \mathbb{F}_2^n \mid A \cdot \mathbf{x} = \mathbf{0}\}$. With linearity of expectation we calculate:

$$\begin{aligned}\mathbb{E}[|\ker(B+P)|] &= \sum_{\mathbf{x} \in \mathbb{F}_2} \Pr[(B+P) \cdot \mathbf{x} = 0] \\ &= \sum_{\mathbf{x} \in \mathbb{F}_2} \Pr[B \cdot \mathbf{x} = P \cdot \mathbf{x}]\end{aligned}$$

Note that $B \cdot \mathbf{x}$ is a fixed vector whereas $P \cdot \mathbf{x}$ is a uniformly distributed over all vectors of weight $|x|$. Thus, the probability that this happens to be $B \cdot \mathbf{x}$ is exactly $\binom{n}{|\mathbf{x}|}^{-1}$ if $|B \cdot \mathbf{x}| = |\mathbf{x}|$ and 0 otherwise. Thus the above is at most

$$\sum_{w=0}^{n} \sum_{\mathbf{x} \in \mathbb{F}_2^n : |\mathbf{x}| = w} \binom{n}{w}^{-1} = n+1 \ . \qquad \blacktriangleleft$$

We set $A := B + P$. By Markov's inequality, $|\ker(A)| \leq n^2$ with high probability, and therefore also $\operatorname{rank}(A) \geq n - 2\log(n)$ with high probability. Also, each row of A has Hamming weight at most $k+1$.

It remains to show that A has the desired expansion properties. First we fix a set U of size t and a well-increasing sequence $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ and estimate the probability that $|\mathbf{u}_i \cdot A_U| \leq w$ for all i. For this we need the following fact about random walks in the Hamming cube which we will prove later.

▶ **Lemma 23** (Hamming Cube Mixing Lemma). *Let $U \subseteq [n]$ and $\mathbf{z} \in \{0,1\}^U$. Let $\mathbf{x}$ be the endpoint of a length d random walk in $\{0,1\}^n$ starting at $\mathbf{0}$. Then*

$$\Pr[\mathbf{x}_U = \mathbf{z}] \leq 2\left(\frac{1+(1-2/n)^d}{2}\right)^{|U|}.$$

In particular if $d \geq n$ and $|U| = t$ is sufficiently large, then this probability is at most $2^{-2t/3}$. From this lemma it is easy to show the following:

▶ **Lemma 24.** *Let $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ be a well-increasing sequence. Then the probability that $|\mathbf{u}_i \cdot A_U| \leq w$ for all i is at most $2^{-\frac{2\cdot\ell\cdot t}{3}} \cdot \binom{t}{\leq w}^\ell$.*

Proof. Let $\mathcal{E}_j$ be the event that $|\mathbf{u}_i \cdot A_U| \leq 2n/k$ for all $1 \leq i \leq j$. We want to bound $\Pr[\mathcal{E}_\ell] = \prod_{j=1}^{\ell} \Pr[|\mathbf{u}_j \cdot A_U| \leq 2n/k \mid \mathcal{E}_{j-1}]$. We claim that for each $1 \leq j \leq \ell$ the probability $\Pr[|\mathbf{u}_j \cdot A_U| \leq 2n/k \mid \mathcal{E}_{j-1}]$ is at most $2^{-2t/3} \cdot \binom{t}{\leq 2n/k}$.

We divide $\mathbf{u}_i$ into an "old part" v_i and a "new part" w_i. Formally, we write $\mathbf{v}_i = \mathbf{u}_i \cap (\mathbf{u}_1 \cup \ldots \mathbf{u}_{i-1})$ and $\mathbf{w}_i = \mathbf{u}_i \setminus (\mathbf{u}_1 \cup \ldots \mathbf{u}_{i-1})$. We know that $|\mathbf{w}_i| \geq n/k$ since the sequence is well-increasing. Also, $\mathbf{u}_i = \mathbf{v}_i + \mathbf{w}_i$. Let $\mathbf{y} \in \mathbb{F}_2^t$ be a fixed vector. Note that

$$\Pr[\mathbf{u}_j \cdot A_U = \mathbf{y} \mid \mathcal{E}_{j-1}] = \Pr[\mathbf{w}_j \cdot A_U = \mathbf{v}_j \cdot A_U + \mathbf{y} \mid \mathcal{E}_{j-1}]$$

Now $\mathbf{v}_j \cdot A_U$ and $\mathcal{E}_{j-1}$ both only depend on the rows $\mathbf{a}_h$ of A with $h \in \mathbf{v}_j$, and $\mathbf{w}_j \cdot A_U$ is independent these. Thus, it suffices to bound $\Pr[\mathbf{w}_j \cdot A_U = \mathbf{z}]$ for some unknown but fixed vector $\mathbf{z}$. Remember that $A = B + P$ where P is a random $n \times n$ permutation matrix.

$$\Pr[\mathbf{w}_j \cdot A_U = \mathbf{z}] = \Pr[\mathbf{w}_j \cdot B_U = \mathbf{z} + P_U \cdot \mathbf{w}_j]$$

What is the distribution of $\mathbf{w}_j \cdot B_U$? It is the sum of $|\mathbf{w}_j|$ rows of B_U and thus distributed like the endpoint of a $|\mathbf{w}_j| \cdot k \geq n$ step random walk in $\{0,1\}^n$ starting at $\mathbf{0}$ and then projected to the coordinates in U. By the Hamming Cube Mixing Lemma (Lemma 23) with $d = n$ we get

$$\Pr[\mathbf{w}_j \cdot A_U = \mathbf{z}] \leq 2\left(\frac{1+\left(1-\frac{2}{n}\right)^n}{2}\right)^t \leq 2^{-2t/3}.$$

We conclude that $\Pr[\mathbf{u}_j \cdot A_U = \mathbf{y} \mid \mathcal{E}_{j-1}] \leq 2^{-2t/3}$ for every fixed $\mathbf{y} \in \mathbb{F}_2^t$. Therefore

$$\Pr[|\mathbf{u}_j \cdot A_U| \leq 2n/k \mid \mathcal{E}_{j-1}] \leq 2^{-2t/3} \cdot \binom{t}{\leq 2n/k}.$$

This proves the claim. Via the chain rule, the claim immediately implies the lemma. ◀

To prove the theorem, it remains to do a union bound over the choices of $U \subseteq [n]$ and the well-increasing sequence. The number of ways to choose $U \subseteq [n]$ of size t is $\binom{n}{t} \leq \left(\frac{en}{t}\right)^t \leq k^t$. Bounding the number of well-increasing sequences is more subtle.

▶ **Lemma 25.** *The number of well-increasing sequences is at most $k^{\frac{4\ell n}{k}} \cdot 2^{\frac{4\ell^2}{k} \cdot n}$.*

Proof. First, write $\mathbf{u} := \mathbf{u}_1 \cup \cdots \cup \mathbf{u}_\ell$ and note that $|\mathbf{u}| \leq \frac{4\ell n}{k}$. Thus, the number of possible $\mathbf{u}$ is at most $\binom{n}{\leq \frac{4\ell n}{k}} \leq k^{\frac{4\ell n}{k}}$ Once we have chosen $\mathbf{u}$, there are at most $2^{|\mathbf{u}|}$ choices for each individual $\mathbf{u}_i$ and at most $2^{\ell \cdot |\mathbf{u}|} \leq 2^{\frac{4\ell^2}{k} \cdot n}$ well-increasing sequences. ◀

Let us now multiply (1) the number of choices for U, (2) the number of well-increasing sequences, and (3) for a fixed U and well-increasing sequence $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$, the probability that $|\mathbf{u}_i \cdot A_U| \leq w$. We see that this is at most

$$k^t \cdot k^{\frac{4\ell n}{k}} \cdot 2^{\frac{4\ell^2}{k} \cdot n} \cdot 2^{-\frac{2 \cdot \ell \cdot t}{3}} \cdot \binom{t}{\leq w}^\ell$$
$$= 2^{\frac{\log^2(k) \cdot n}{k} \cdot \left(60 + 4 \cdot 5 + 4 \cdot 5^2 - \frac{2}{3} 5 \cdot 60 + 2 \cdot 5\right)} = o(1) \ .$$

Here we used $\binom{t}{\leq w} \leq \left(\frac{et}{w}\right)^w \leq k^{2n/k}$. We conclude that A has the desired expansion properties with high probability. In addition, it has rank at least $n - 2\log(n)$, and every row has Hamming weight at most $k + 1$. This concludes the proof. ◀

5 Proof of Lemma 23

Let Q be the random walk matrix of the n-dimensional Hamming cube. That is, $Q_{\mathbf{x},\mathbf{y}} = 1/n$ if $\mathbf{x}$ and $\mathbf{y}$ have Hamming distance 1, and 0 otherwise. Note that Q is a $(2^n \times 2^n)$-matrix, i.e., it takes as input vectors of dimension 2^n, or equivalently, functions from $\mathbb{F}_2^n$ to $\mathbf{R}$. If $f : \mathbb{F}_2^n \to [0, 1]$ is a probability distribution over $\mathbb{F}_2^n$, then $Q^d f$ is the distribution that we get when sampling $\mathbf{x} \sim f$ and performing a random walk of length d. Let f be the function that is 1 at $\mathbf{0}$ and 0 elsewhere. For $\mathbf{X}$ being the endpoint of an d-step random walk starting at $\mathbf{0}$, it holds that

$$\Pr[\mathbf{X} = \mathbf{y}] = (Q^t f)(\mathbf{y}) \ .$$

Fortunately, we can understand $Q^t f$, since we know the eigenvalues of Q: The Hamming cube is the Cayley graph of the additive group of $\mathbb{F}_2^n$ with generating set $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$. The reader who could not make sense of this last sentence may read the next couple of paragraphs. The reader who is familiar with Cayley graphs and the discrete Fourier transform can skip them.

▶ **Definition 26.** For $S \subseteq [n]$, define $\chi_S : \mathbb{F}_2^n \to \mathbb{R}$ by

$$\chi_S(\mathbf{x}) := (-1)^{\sum_{i \in S} x_i} \ .$$

One checks that the χ_S form an orthonormal basis of the space of functions $\mathbb{F}_2^n \to \mathbb{R}$ when we choose the following inner product:

$$\langle f, g \rangle := \mathop{\mathbb{E}}_{\mathbf{x} \in \mathbb{F}_2^n} [f(\mathbf{x}) g(\mathbf{x})] \ .$$

Each χ_S is an eigenvector of Q:

$$\begin{aligned}
(Q \cdot \chi_S)(\mathbf{x}) &= \sum_{\mathbf{y}} Q_{\mathbf{x},\mathbf{y}} \chi_S(\mathbf{y}) = \sum_{\mathbf{y} : d_H(\mathbf{x},\mathbf{y}) = 1} \frac{1}{n} \chi_S(\mathbf{y}) \\
&= \sum_{i=1}^n \frac{1}{n} \chi_S(\mathbf{x} + \mathbf{e}_i) = \sum_{i=1}^n \frac{1}{n} \chi_S(\mathbf{x}) \chi_S(\mathbf{e}_i) \\
&= \chi_S(\mathbf{x}) \frac{1}{n} \sum_{i=1}^n \chi_S(\mathbf{e}_i) \ .
\end{aligned}$$

So $\lambda_S := \frac{1}{n}\sum_{i=1}^n \chi_S(\mathbf{e}_i)$ is the eigenvector of χ_S. Let us evaluate λ_S:

$$\lambda_S \quad = \quad \frac{1}{n}\sum_{i=1}^n \chi_S(\mathbf{e}_i) = \frac{1}{n}\sum_{i=1}^n (-1)^{[i\in S]} = \frac{1}{n}(n-2|S|) = 1 - \frac{2|S|}{n} \ .$$

Let $f : \mathbb{F}_2^n \to \mathbb{R}$ be the function that is 1 on $\mathbf{0}$ and 0 otherwise. To understand $Q^t f$, we write f in the basis of the eigenvectors of Q. Since the χ_S are orthonormal under the scalar product $\langle \cdot, \cdot \rangle$, we can write

$$f = \sum_{S\subseteq[n]} \hat{f}_S \chi_S \ ,$$

where the coefficients $\hat{f}_S$ are

$$\hat{f}_S := \langle f, \chi_S \rangle = \mathop{\mathbb{E}}_{\mathbf{x}\in\mathbb{F}_2^n}[f(\mathbf{x})\chi_S(\mathbf{x})] = 2^{-n},$$

since $\mathbf{x} = \mathbf{0}$ is the only element that contributes to the expectation. Thus,

$$Q^t f \quad = \quad Q^t\left(\sum_S \hat{f}_S\chi_S\right) = Q^t\left(\sum_S 2^{-n}\chi_S\right) = 2^{-n}\sum_S \lambda_S^t \chi_S \ .$$

For $\mathbf{y} \in \mathbb{F}_2^n$, let us bound the probability $\Pr[\mathbf{X} = \mathbf{y}]$: With the above equation, we get

$$\begin{aligned}
(Q^t f)(\mathbf{y}) &= 2^{-n}\sum_S \lambda_S^t \chi_S(\mathbf{y}) = 2^{-n}\sum_S \left(1 - \frac{2|S|}{n}\right)^t \chi_S(\mathbf{y}) \\
&\le 2^{-n}\sum_{s=0}^n \binom{n}{s}\left|1-\frac{2s}{n}\right|^t \quad \text{(since } |\chi_S(\mathbf{y})| = 1\text{)} \\
&\le 2^{-n}\sum_{s=0}^{\lfloor n/2\rfloor}\binom{n}{s}\left|1-\frac{2s}{n}\right|^t + 2^{-n}\sum_{s=\lceil n/2\rceil}^{n}\binom{n}{s}\left|1-\frac{2s}{n}\right|^t \\
&= 2^{-n}\sum_{s=0}^{\lfloor n/2\rfloor}\binom{n}{s}\left|1-\frac{2s}{n}\right|^t + 2^{-n}\sum_{r=0}^{\lfloor n/2\rfloor}\binom{n}{n-r}\left|1-\frac{2(n-r)}{n}\right|^t \\
&= 2\cdot 2^{-n}\sum_{s=0}^{\lfloor n/2\rfloor}\binom{n}{s}\left(1-\frac{2s}{n}\right)^t \\
&\le 2\cdot 2^{-n}\sum_{s=0}^{\lfloor n/2\rfloor}\binom{n}{s}\left(1-\frac{2}{n}\right)^{st} \le 2\cdot 2^{-n}\sum_{s=0}^{n}\binom{n}{s}\left(1-\frac{2}{n}\right)^{st} \\
&= 2\left(\frac{1+\left(1-\frac{2}{n}\right)^t}{2}\right)^n \ .
\end{aligned}$$

This proves the lemma for $U = [n]$. In general, however, we are interested in the distribution of $\mathbf{X}_U$, i.e., $\mathbf{X}$ projected to the coordinates in U.

▶ **Observation 27.** *Perform a "lazy" random walk on $\{0,1\}^{|U|}$ as follows: Start at $\mathbf{0}$. At each step, take each edge with probability $1/n$. With the remaining probability $1 - |U|/n$, don't move in this step. Then the end point of this walk after t steps has distribution $\mathbf{X}_U$.*

Let Q be transition matrix of the random walk on $\{0,1\}^{|U|}$. Then

$$\tilde{Q} := \frac{|U|}{n}Q + \frac{n-|U|}{n}I$$

is the transition matrix of the lazy random walk described above. For each $S \subseteq U$, χ_S is an eigenvector of Q, and the corresponding eigenvalue is $\lambda_S = 1 - \frac{2|S|}{|U|}$. The matrix $\tilde{Q}$ has the same eigenvectors as Q, and its eigenvalues are

$$\tilde{\lambda}_S = \frac{|U|}{n}\lambda_S + \frac{n-|U|}{n} \cdot 1 = 1 - \frac{2|S|}{n} \ .$$

Let $f : \{0,1\}^{|U|} \to \mathbb{R}$ be the function that is 1 at $\mathbf{0}$ and 0 elsewhere. We write $f = \sum_{S \subseteq U} \hat{f}_S \chi_S$. Let $u := |U|$. By the same calculation as above, $\hat{f}_S = 2^{-u}$. Thus, for $\mathbf{y} \in \{0,1\}^u$ we get

$$\begin{aligned}(\tilde{Q}^t f)(\mathbf{y}) &= 2^{-u} \sum_S \lambda_S^t \chi_S(\mathbf{y}) = 2^{-u} \sum_S \left(1 - \frac{2|S|}{n}\right)^t \chi_S(\mathbf{y}) \\ &\leq 2^{-u} \sum_{s=0}^{u} \binom{u}{s} \left|1 - \frac{2s}{n}\right|^t \quad (\text{siuce } |\chi_S(\mathbf{y})| = 1)\,.\end{aligned}$$

If $u \leq n/2$, we observe that all eigenvalues $1 - 2s/n$ are non-negative.[2] In this case we continue:

$$2^{-u} \sum_{s=0}^{u} \binom{u}{s} \left(1 - \frac{2s}{n}\right)^t \leq 2^{-u} \sum_{s=0}^{u} \binom{u}{s} \left(1 - \frac{2}{n}\right)^{st} = \left(\frac{1 + \left(1 - \frac{2}{n}\right)^t}{2}\right)^u \tag{1}$$

and we are done. If $u > n/2$, things get more tricky. We split the sum in two parts:

$$2^{-u} \sum_{s=0}^{\lfloor n/2 \rfloor} \binom{u}{s} \left(1 - \frac{2s}{n}\right)^t + 2^{-u} \sum_{s=\lfloor n/2 \rfloor + 1}^{u} \binom{u}{s} \left(\frac{2s}{n} - 1\right)^t \ . \tag{2}$$

We can bound the first sum exactly similar as in (1):

$$\begin{aligned}2^{-u} \sum_{s=0}^{\lfloor n/2 \rfloor} \binom{u}{s} \left(1 - \frac{2s}{n}\right)^t &\leq 2^{-u} \sum_{s=0}^{\lfloor n/2 \rfloor} \binom{u}{s} \left(1 - \frac{2}{n}\right)^{st} \leq 2^{-u} \sum_{s=0}^{u} \binom{u}{s} \left(1 - \frac{2}{n}\right)^{st} \\ &= \left(\frac{1 + \left(1 - \frac{2}{n}\right)^t}{2}\right)^u \ .\end{aligned}$$

Let us bound the second sum in (2). For notational convenience, we let it run from $\lceil n/2 \rceil$ to u, only making it larger. We change the parameter s to $r := u - s$. Thus

$$\begin{aligned}2^{-u} \sum_{s=\lceil n/2 \rceil}^{u} \binom{u}{s} \left(\frac{2s}{n} - 1\right)^t &= 2^{-u} \sum_{r=0}^{u-\lceil n/2 \rceil} \binom{u}{u-r} \left(\frac{2(u-r)}{n} - 1\right)^t \\ &= 2^{-u} \sum_{r=0}^{u-\lceil n/2 \rceil} \binom{u}{r} \left(\frac{2u-n}{n} - \frac{2r}{n}\right)^t \\ &\leq 2^{-u} \sum_{r=0}^{u-\lceil n/2 \rceil} \binom{u}{r} \left(1 - \frac{2r}{n}\right)^t \qquad (\text{since } u \leq n) \\ &\leq 2^{-u} \sum_{r=0}^{u-\lceil n/2 \rceil} \binom{u}{r} \left(1 - \frac{2}{n}\right)^{rt} \\ &\leq 2^{-u} \sum_{r=0}^{u} \binom{u}{r} \left(1 - \frac{2}{n}\right)^{rt} = \left(\frac{1 + \left(1 - \frac{2}{n}\right)^t}{2}\right)^u \ .\end{aligned}$$

Thus, both sums in (2) are bounded by (1) and the lemma follows.

[2] The reader might observe that in our application indeed $|U| \ll n/2$.

References

1. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58. ACM, 2015. URL: http://dl.acm.org/citation.cfm?id=2746539, doi:10.1145/2746539.2746612.
2. Eli Ben-Sasson and Russell Impagliazzo. Random cnf's are hard for the polynomial calculus. *Computational Complexity*, 19(4):501–519, 2010. doi:10.1007/s00037-010-0293-1.
3. Shiteng Chen, Dominik Scheder, Navid Talebanfard, and Bangsheng Tang. Exponential lower bounds for the PPSZ k-SAT algorithm. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1253–1263. SIAM, 2013. doi:10.1137/1.9781611973105.91.
4. Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon M. Kleinberg, Christos H. Papadimitriou, Prabhakar Raghavan, and Uwe Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-sat based on local search. *Theor. Comput. Sci.*, 289(1):69–83, 2002. doi:10.1016/S0304-3975(01)00174-8.
5. Paul Erdős and Horst Sachs. Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl. (Regular graphs with given girth and minimal number of knots.). *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg, Math.-Naturwiss.*, 12:251–258, 1963.
6. Timon Hertli. 3-sat faster and simpler – unique-sat bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014. doi:10.1137/120868177.
7. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
8. Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.
9. Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. doi:10.1007/BF02126799.
10. Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.
11. Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.*, 1999, 1999. URL: http://cjtcs.cs.uchicago.edu/articles/1999/11/contents.html.
12. Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS'99, 17-18 October, 1999, New York, NY, USA*, pages 410–414. IEEE Computer Society, 1999. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6604, doi:10.1109/SFFCS.1999.814612.
13. G. S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, (2):115–125, 1968.
14. Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.

Subspace Designs Based on Algebraic Function Fields*

Venkatesan Guruswami[†1], Chaoping Xing[‡2], and Chen Yuan[3]

1 **Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA**
guruswami@cmu.edu

2 **School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore**
xingcp@ntu.edu.sg

3 **School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore**
yuan0064@e.ntu.edu.sg

Abstract

Subspace designs are a (large) collection of high-dimensional subspaces $\{H_i\}$ of $\mathbb{F}_q^m$ such that for any low-dimensional subspace W, only a small number of subspaces from the collection have non-trivial intersection with W; more precisely, the sum of dimensions of $W \cap H_i$ is at most some parameter L. The notion was put forth by Guruswami and Xing (STOC'13) with applications to list decoding variants of Reed-Solomon and algebraic-geometric codes, and later also used for explicit rank-metric codes with optimal list decoding radius.

Guruswami and Kopparty (FOCS'13, Combinatorica'16) gave an explicit construction of subspace designs with near-optimal parameters. This construction was based on polynomials and has close connections to folded Reed-Solomon codes, and required large field size (specifically $q \geqslant m$). Forbes and Guruswami (RANDOM'15) used this construction to give explicit constant degree "dimension expanders" over large fields, and noted that subspace designs are a powerful tool in linear-algebraic pseudorandomness.

Here, we construct subspace designs over any field, at the expense of a modest worsening of the bound L on total intersection dimension. Our approach is based on a (non-trivial) extension of the polynomial-based construction to algebraic function fields, and instantiating the approach with cyclotomic function fields. Plugging in our new subspace designs in the construction of Forbes and Guruswami yields dimension expanders over $\mathbb{F}^n$ for any field $\mathbb{F}$, with logarithmic degree and expansion guarantee for subspaces of dimension $\Omega(n/(\log\log n))$.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems, F.2.2 Nonnumerical Algorithms and Problems, G.1.3 Numerical Linear Algebra

Keywords and phrases Pseudorandomness, algebraic codes, explicit constructions, expanders, linear algebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.86

* A full version of the paper is available at https://arxiv.org/abs/1704.05992.
† Research supported in part by NSF CCF-1422045.
‡ Research supported in part by the Singapore MoE Tier 1 grants RG20/13 and RG25/16.

1 Introduction

An emerging theory of "linear-algebraic pseudorandomness" studies the linear-algebraic analogs of fundamental Boolean pseudorandom objects where the rank of subspaces plays the role of the size of subsets. A recent work [4] studied the interrelationships between several such algebraic objects such as subspace designs, dimension expanders, rank condensers, and rank-metric codes, and highlighted the fundamental unifying role played by *subspace designs* in this web of connections.

Informally, a subspace design is a collection of subspaces of a vector space $\mathbb{F}_q^m$ (throughout we denote by $\mathbb{F}_q$ the finite field with q elements) such that any low-dimensional subspace W intersects only a small number of subspaces from the collection. More precisely:

▶ **Definition 1.** A collection $H_1, H_2, \ldots, H_M$ of b-dimensional subspaces of $\mathbb{F}_q^m$ form an (s, L)-(strong) subspace design, if for every s-dimensional subspace $W \subset \mathbb{F}_q^m$, $\sum_{i=1}^{M} \dim(W \cap H_i) \leqslant L$.

In particular, this implies that at most L subspaces H_i have non-trivial intersection with W. A collection meeting this weaker requirement is called a *weak* subspace design; unless we mention otherwise, by subspace design we always mean a strong subspace design in this paper. One would like the dimension b of each subspace in the subspace design to be large, typically $\Omega(m)$ for applications of interest, L to be small, and the number of subspaces M to be large.

Subspace designs were introduced by the first two authors in [11], where they used them to improve the list size and efficiency of list decoding algorithms for algebraic-geometric codes, yielding efficiently list-decodable codes with optimal redundancy over fixed alphabets and small output list size. A standard probabilistic argument shows that a random collection of subspaces forms a good subspace design with high probability. Subsequently, Guruswami and Kopparty [7] gave an explicit construction of subspace designs, nearly matching the parameters of random constructions, albeit over large fields.

Intriguingly, the construction in [7] was based on algebraic list-decodable codes (specifically folded Reed-Solomon codes). Recall that improving the list-decodability of such codes was the motivation for the formulation of subspace designs in the first place! This is yet another compelling example of the heavily intertwined nature of error-correcting codes and other pseudorandom objects. The following states one of the main trade-offs achieved by the construction in [7].

▶ **Theorem 2** (Folded Reed-Solomon based construction [7]). *For every $\varepsilon \in (0, 1)$, positive integers s, m with $s \leqslant \varepsilon m/4$, and a prime power $q > m$, there exists an explicit*[1] *collection of $M = q^{\Omega(\varepsilon m/s)}$ subspaces in $\mathbb{F}_q^m$, each of dimension at least $(1 - \varepsilon)m$, which form a $(s, \frac{2s}{\varepsilon})$-(strong) subspace design.*

Note the requirement of the field size q being larger than the ambient dimension m in their construction. To construct subspace designs over small fields, they use a construction over a large extension field $\mathbb{F}_{q^r}$, and view b-dimensional subspaces of $\mathbb{F}_{q^r}^{m'}$ as br-dimensional subspaces of $\mathbb{F}_q^{rm'}$. However, this transformation need not preserve the "strongness" of the subspace design, and an (s, L)-subspace design over the extension field only yields an (s, L)-weak subspace design over $\mathbb{F}_q$.

[1] By explicit, we mean a deterministic construction that runs in time $\text{poly}(q, m, M)$ and outputs a basis for each of the subspaces in the subspace design.

The strongness property is crucial for all the applications of subspace designs in [4]. In particular, the strongness is what drives the construction of dimension expanders (defined below) of low degree. The weak subspace design property does *not* suffice for these applications.

▶ **Definition 3.** A collection of linear maps $A_1, A_2, \ldots, A_d : \mathbb{F}^n \to \mathbb{F}^n$ is said to be a (b, α)-dimension expander if for every subspace V of $\mathbb{F}^n$ of dimension at most b, $\dim(\sum_{i=1}^{d} A_i(V)) \geqslant (1+\alpha)\cdot\dim(V)$. The number of maps d is the "degree" of the expander, and α is the expansion factor.

Using the subspace designs constructed in Theorem 2 in a black-box fashion, Forbes and Guruswami [4] gave explicit $(\Omega(n), \Omega(1))$-dimension expanders of $O(1)$ degree when $|\mathbb{F}| \geqslant \text{poly}(n)$. Here explicit means that the maps A_i are specified explicitly, say by the matrix representing their action with respect to some fixed basis. Extending Theorem 2 to smaller fields will yield constant-degree $(\Omega(n), \Omega(1))$-dimension expanders over all fields. The only known constructions of such dimension expanders over finite fields rely on monotone expanders [3, 2], a rather complicated (and remarkable) form of bipartite vertex expanders whose neighborhood maps are monotone. Even the existence of constant-degree monotone expanders does not follow from standard probabilistic methods, and the only known explicit construction is a sophisticated one using the group $\text{SL}_2(\mathbb{R})$ by Bourgain and Yehudayoff [1]. (Earlier, Dvir and Shpilka [2] constructed monotone expanders of logarithmic degree using Cayley graphs over the cyclic group, yielding logarithmic degree $(\Omega(n), \Omega(1))$-dimension expanders.)

In light of this, it is a very interesting question to remove the field size restriction in Theorem 2 above, as it will yield an arguably simpler construction of constant-degree dimension expanders over every field, and which might also offer a quantitatively better trade-off between the degree and expansion factor. We note that probabilistic constructions achieve similar parameters (in fact a slightly larger sized collection with $q^{\Omega(\varepsilon m)}$ subspaces) with no restriction on the field size (one can even take $q = 2$).

Our construction. The large field size in Theorem 2 was inherited from Reed-Solomon codes, which are defined over a field of size at least the code length. Our main contribution in this work is a construction of subspace designs based on algebraic function fields, which permits us to construct subspace designs over small fields. By instantiating this approach with a construction based on cyclotomic function fields, we are able to prove the following main result in this work:

▶ **Theorem 4** (Main Theorem). *For every $\varepsilon \in (0, 1)$, a prime power q and positive integers s, m such that $s \leq \varepsilon m/4$, there exists an explicit construction of $M = \Omega(q^{\lfloor \varepsilon m/(2s) \rfloor}/\varepsilon)$ subspaces in $\mathbb{F}_q^m$, each of dimension at least $(1-\varepsilon)m$, which form an $\left(s', \frac{2s'\lceil \log_q(m) \rceil}{\varepsilon}\right)$-strong subspace design for all $s' \leqslant s$.*

Note that we state a slightly stronger property that the bound on intersection size improves for subspaces of lower dimension $s' \leqslant s$. This property also holds for Theorem 2 and in fact is important for the dimension expander construction in [4], and so we make it explicit.

The bound on intersection size we guarantee above is worse than the one from the random construction by a factor of $\log_q m$. The result of Theorem 2 can be viewed as a special case of Theorem 4 since $\log_q m \leqslant 1$ when $q > m$. The factor $\log_q m$ comes out as a trade-off of the explicit construction vs the random construction given in [11]. The extension field based construction using Theorem 2 would yield an $(s, O(s^2/\varepsilon))$-subspace design (since an

(s, L)-weak subspace design is trivially an (s, sL)-(strong) subspace design). The bound we achieve is better for all $s = \Omega(\log_q m)$. In the use of subspace designs in the dimension expander construction of [4], s governs the dimension of the subspaces which are guaranteed to expand, which we would like to be large (and ideally $\Omega(m)$). The application of subspace designs to list decoding [11, 9] employs the parameter choice $m = O(s)$ in order keep the alphabet size q^m small. Therefore, our improvement applies to a meaningful setting of parameters that is important for the known applications of (strong) subspace designs.

Application to dimension expanders over small fields. By plugging in the subspace designs of Theorem 4 into the dimension expander construction of [4], we can get the following:

▶ **Theorem 5.** *For every prime power q and positive integer $n \geqslant q$, there exists an explicit construction of a $\left(b = \Omega\left(\frac{n}{\log_q \log_q n}\right), 1/3\right)$-dimension expander with $O(\log_q n)$ degree.*

For completeness, let us very quickly recap how such dimension expanders may be obtained from the subspace designs of Theorem 4, using the "tensor-then-condense" approach in [4]. We begin with linear maps $T_1, T_2 : \mathbb{F}^n \to \mathbb{F}^{2n}$, where $T_1(v) = (v; 0)$ and $T_2(v) = (0; v)$ – these trivially achieve expansion factor 2 by doubling the ambient dimension. Then we take the subspace design of Theorem 4 with $m = 2n$, $\varepsilon = 1/2$, $s = 2b$, and $M = 12\lceil \log_q m \rceil$ subspaces H_i (if $b = \beta n/(\log_q \log_q n)$ for small enough absolute constant $\beta > 0$, Theorem 4 guarantees these many subspaces). Let $E_i : \mathbb{F}^{2n} \to \mathbb{F}^n$ be linear maps such that $H_i = \ker(E_i)$. The dimension expander consists of the $2M$ composed maps $E_i \circ T_j$ for $i = 1, 2, \ldots, M$ and $j = 1, 2$. Briefly, the analysis of the expansion in dimension proceeds as follows. Let V be a subspace of $\mathbb{F}^n$ with $\dim(V) = \ell \leqslant b$, and let $W = T_1(V) + T_2(V)$ be the 2ℓ-dimensional subspace of $\mathbb{F}^{2n}$ after the tensoring step. The strong subspace design property implies that the number of maps E_i for which $\dim(E_i W) < 4\ell/3$ – which is equivalent to $\dim(W \cap H_i) > 2\ell/3$ – is less than $12\lceil \log_q m \rceil = M$. So there must be an i for which $\dim(E_i W) \geqslant 4\ell/3$, and this E_i when composed with T_1 and T_2 will expand V to a subspace of dimension at least $\frac{4}{3}\dim(V)$.

By using a method akin to the conversion of Reed-Solomon codes over extension fields to BCH codes over the base field, applied to the large field subspace designs of Theorem 2, Forbes and Guruswami [4] constructed $(\Omega(n/\log n), \Omega(1))$-dimension expanders of $O(\log n)$ degree. In contrast, our construction here guarantees expansion for dimension up to $\Omega(n/(\log \log n))$. The parameters offered by Theorem 5 are, however, weaker than both the construction given in [2], which has logarithmic degree but expands subspaces of dimension $\Omega(n)$, as well as the one in [1], which further gets constant degree. However, we do not go through monotone expanders which are harder to construct than vertex expanders, and our construction works fully within the linear-algebraic setting. We hope that the ideas in this work pave the way for a subspace design similar to Theorem 2 over small fields, and the consequent construction of constant-degree $(\Omega(n), \Omega(1))$-dimension expanders over all fields. In fact, all that is required for this is an $(s, O(s))$-subspace design with a sufficiently large constant number of subspaces, each of dimension $\Omega(m)$.

Construction approach. The generalization of the polynomials-based subspace design from [7] to take advantage of more general algebraic function fields is not straightforward. The natural approach would be to replace the space of low-degree polynomials by a Riemann-Roch space consisting of functions of bounded pole order ℓ at some place. We prove that such a construction can work, provided the degree ℓ is less than the degree of the field extension (and some other mild condition is met). However, this degree restriction is a severe one, and the dimension of the associated Riemann-Roch space will typically be too small (as

the "genus" of the function field, which measures the degree minus dimension "defect," will be large), unless the field size is large. Therefore, we don't know an instantiation of this approach that yields a family of good subspace designs over a fixed size field.

Let us now sketch the algebraic crux of the polynomial based construction in [7], and the associated challenges in extending it to other function fields. The core property of a dimension s subspace W of polynomials underlying the construction of Theorem 2 is the following: If $f_1, f_2, \ldots, f_s \in \mathbb{F}_q[X]$ of degree less than $q-1$ are linearly independent over $\mathbb{F}_q$ (these s polynomials being a basis of the subspace W), then the "folded Wronskian," which is the determinant of the matrix $M(f_1, f_2, \ldots, f_s)$ whose i,j'th entry is $f_j(\gamma^{i-1}X)$, is a *nonzero* polynomial in $\mathbb{F}_q[X]$. Here γ is an arbitrary primitive element of $\mathbb{F}_q$. One might compare this with the classical Wronskian criterion for linear dependence over characteristic zero fields (and also holds when characteristic is bigger than the degree of the f_i's), based on the singularity of the $s \times s$ matrix whose i,j'th entry is $\frac{d^{i-1} f_j}{dX^{i-1}}$.

One approach is to prove this claim about the folded Wronskian is via a "list size" bound from list decoding: one can prove that for any $A_1, \ldots, A_s \in \mathbb{F}_q[X]$, not all 0, the space of solutions $f \in \mathbb{F}_q[X]_{<(q-1)}$ to

$$A_1(X)f(X) + A_2(X)f(\gamma X) + \cdots + A_s(X)f(\gamma^{s-1}X) = 0 \tag{1}$$

has dimension at most $s-1$. (This was the basis of the linear-algebraic list decoding algorithm for folded Reed-Solomon codes [6, 8].) Stating the contrapositive, if $f_1, f_2, \ldots, f_s$ are linearly independent over $\mathbb{F}_q[X]$, then the rows of the matrix $M(f_1, f_2, \ldots, f_s)$ are linearly independent, and therefore its determinant, the folded Wronskian, is a nonzero polynomial. On the other hand, being the determinant of an $s \times s$ matrix whose entries are degree m polynomials, the folded Wronskian has degree at most ms. To prove the subspace design property, one then establishes that for each subspace H_i in the collection that intersects $W = \mathrm{span}(f_1, \ldots, f_s)$, the determinant picks up a number of distinct roots each with $\dim(W \cap H_i)$ multiplicity, the set of roots for different intersecting H_i being disjoint from each other. The total intersection bound then follows because the folded Wronskian has at most ms roots, counting multiplicities.

One can try to mimic the above approach for folded algebraic-geometric (AG) codes, with f^σ for some suitable automorphism σ playing the role of the shifted polynomial $f(\gamma X)$. This, however, runs into significant trouble, as the bound on number of solutions f to the functional equation analogous to (1), $A_1 f + A_2 f^\sigma + \cdots + A_s f^{\sigma^{s-1}} = 0$, is much higher. The list of solutions is either exponentially large and needs pruning via pre-coding the folded AG codes with subspace-evasive sets [10], or it is much bigger than q^{s-1} in the constructions based on cyclotomic function fields and narrow ray class fields where the folded AG codes work directly [5, 12].

Let F/K be a function field where the extension is Galois with Galois group generated by an automorphism σ. We choose the m-dimensional ambient space $\mathcal{V} \cong \mathbb{F}_q^m$ to be a carefully chosen subspace of a Riemann-Roch space in F of degree $\ell \gg m$ (specifically, we require $\ell \geqslant m + 2\mathfrak{g}$ where $\mathfrak{g}$ is the genus). We then establish that if $f_1, f_2, \ldots, f_s \in \mathcal{V}$ are linearly independent over $\mathbb{F}_q$, a certain "automorphism Moore matrix" $M_\sigma(f_1, f_2, \ldots, f_s)$ is non-singular. The determinant of this Moore matrix is thus a non-zero function in F, and this generalizes the folded Wronskian criterion for polynomials mentioned above.

This non-singularity result is proved in two steps. First, we show that for functions in $\mathcal{V}$, linear independence over $\mathbb{F}_q$ implies linear independence over K. Then we show that for any $f_1, \ldots, f_s \in F$ that are linearly independent over $K = F^\sigma$, the automorphism Moore matrix associated with σ is non-singular. With our hands on the non-zero function

$\Delta = \mathsf{det}(M_\sigma(f_1, f_2, \ldots, f_s))$, we can proceed as in the folded Reed-Solomon case – the part about Δ picking up many zeroes whenever a subspace in the collection intersects $\mathrm{span}(f_1, \ldots, f_s)$ also generalizes. The pole order of Δ, however, is now ℓs instead of ms in the polynomial-based construction. This is the cause for the worse bound on total intersection dimension in our Theorem 4. The detailed analysis of the above function field generalization will be presented in a full version of this paper. In the current version, we present only constructions without proof and hence "automorphism Moore matrix" is not introduced.

Organization. We begin with a quick review of background on algebraic function fields in general and cyclotomic function fields in particular in Section 2. We presentour constructions of subspace designs from function fields in Section 3 In Section 4, we instantiate our construction with specific cyclotomic function fields and derive our main consequence for subspace designs and establish Theorem 4.

2 Preliminaries on function fields

Background on function fields. Throughout this paper, $\mathbb{F}_q$ denotes the finite field of q elements. A function field F over $\mathbb{F}_q$ is a field extension over $\mathbb{F}_q$ in which there exists an element z of F that is transcendental over $\mathbb{F}_q$ such that $F/\mathbb{F}_q(z)$ is a finite extension. $\mathbb{F}_q$ is called the full constant field of F if the algebraic closure of $\mathbb{F}_q$ in F is $\mathbb{F}_q$ itself. In this paper, we always assume that $\mathbb{F}_q$ is the full constant field of F, denoted by $F/\mathbb{F}_q$.

Each discrete valuation ν from F to $\mathbb{Z} \cup \{\infty\}$ defines a local ring $O = \{f \in F : \nu(f) \geqslant 0\}$. The maximal ideal P of O is called a *place*. We denote the valuation ν and the local ring O corresponding to P by ν_P and O_P, respectively. The residue class field O_P/P, denoted by F_P, is a finite extension of $\mathbb{F}_q$. The extension degree $[F_P : \mathbb{F}_q]$ is called *degree* of P, denoted by $\deg(P)$.

Let $\mathbb{P}_F$ denote the set of places of F. A divisor D of F is a formal sum $\sum_{P \in \mathbb{P}_F} m_P P$, where $m_P \in \mathbb{Z}$ are equal to 0 except for finitely many P. The degree of D is defined to be $\deg(D) = \sum_{P \in \mathbb{P}_F} m_P \deg(P)$. We say that D is positive, denoted by $D \geqslant 0$, if $m_P \geqslant 0$ for all $P \in \mathbb{P}_F$. For a nonzero function f, the principal divisor (f) is defined to be $\sum_{P \in \mathbb{P}_F} \nu_P(f) P$. Then the degree of the principal divisor (f) is 0. The Riemann-Roch space associated with a divisor D, denoted by $\mathcal{L}(D)$, is defined by

$$\mathcal{L}(D) := \{f \in F \setminus \{0\} : (f) + D \geqslant 0\} \cup \{0\}. \tag{2}$$

Then $\mathcal{L}(D)$ is a finite dimensional space over $\mathbb{F}_q$. By the Riemann-Roch theorem [15], the dimension of $\mathcal{L}(D)$, denoted by $\dim_{\mathbb{F}_q}(D)$, is lower bounded by $\deg(D) - \mathfrak{g} + 1$, i.e., $\dim_{\mathbb{F}_q}(D) \geqslant \deg(D) - \mathfrak{g} + 1$, where $\mathfrak{g}$ is the genus of F. Furthermore, $\dim_{\mathbb{F}_q}(D) = \deg(D) - \mathfrak{g} + 1$ if $\deg(D) \geqslant 2\mathfrak{g} - 1$. In addition, we have the following results [15, Lemma 1.4.8 and Corollary 1.4.12(b)]:

- **(i)** If $\deg(D) < 0$, then $\dim_{\mathbb{F}_q}(D) = 0$;
- **(ii)** For a positive divisor G, we have $\dim_{\mathbb{F}_q}(D) - \dim_{\mathbb{F}_q}(D - G) \leqslant \deg(G)$, i.e., $\dim_{\mathbb{F}_q}(D - G) \geqslant \dim_{\mathbb{F}_q}(D) - \deg(G)$.

Let $\mathrm{Aut}(F/\mathbb{F}_q)$ denote the set of automorphisms of F that fix every element of $\mathbb{F}_q$, i.e.,

$$\mathrm{Aut}(F/\mathbb{F}_q) = \{\tau : \tau \text{ is an automorphism of } F \text{ and } \alpha^\tau = \alpha \text{ for all } \alpha \in \mathbb{F}_q\}.$$

For a place $P \in \mathbb{P}_F$ and an automorphism $\sigma \in \mathrm{Aut}(F/\mathbb{F}_q)$, we denote by P^σ the set $\{f^\sigma : f \in P\}$. Then P^σ is a place and moreover we have $\deg(P^\sigma) = \deg(P)$. The place P^σ is called a conjugate place of P. σ also induces an automorphsim of $\mathrm{Aut}(\mathbb{F}_P/\mathbb{F}_q)$. This

implies that there exists an integer $e \geqslant 0$ such that $\alpha^\sigma = \alpha^{q^e}$ for all $\alpha \in F_P$. σ is called the *Frobenius* of P if $e = 1$, i.e., $\alpha^\sigma = \alpha^q$ for all $\alpha \in F_P$. For a place P and a function $f \in O_P$, we denote by $f(P)$ the residue class of f in F_P. Thus, we have $(f(P))^{q^e} = (f(P))^\sigma = f^\sigma(P^\sigma)$.

Background on cyclotomic function fields. Let x be a transcendental element over $\mathbb{F}_q$ and denote by K the rational function field $\mathbb{F}_q(x)$. Let K^{ac} be an algebraic closure of K. Denote by $\mathbb{F}_q[x]$ the polynomial ring $\mathbb{F}_q[x]$. Let $\mathrm{End}(K^{ac})$ be the ring of homomorphisms from K^{ac} to K^{ac}. We define $\rho_x(z) = z^q + xz$ for all $z \in K^{ac}$. For $i \geqslant 2$, we define $\rho_{x^i}(z) = \rho_x(\rho_{x^{i-1}}(z))$. For a polynomial $p(x) = \sum_{i=0}^n a_i x^i \in \mathbb{F}_q[x]$, we define $\rho_{p(x)}(z) = \sum_{i=0}^n a_i \rho_{x^i}(z)$. For simplicity, we denote $\rho_{p(x)}(z)$ by $z^{p(x)}$. It is easy to see that $z^{p(x)} \in \mathbb{F}_q[x][z]$ is a q-linearized polynomial in z of degree q^d, where $d = \deg(p(x))$.

For a polynomial $p(x) \in \mathbb{F}_q[x]$ of degree d, define the set

$$\Lambda_{p(x)} := \{\alpha \in K^{ac} : \alpha^{p(x)} = 0\}. \tag{3}$$

Then $\Lambda_{p(x)} \simeq \mathbb{F}_q[x]/(p(x))$ is an $\mathbb{F}_q[x]$-module and it has exactly q^d elements. Furthermore, $\Lambda_{p(x)}$ is a cyclic $\mathbb{F}_q[x]$-module. For any generator λ of $\Lambda_{p(x)}$, one has $\Lambda_{p(x)} = \{\lambda^A : A \in \mathbb{F}_q[x]/(p(x))\}$ and λ^A is a generator of $\Lambda_{p(x)}$ if and only if $\gcd(A, p(x)) = 1$. The extension $K(\lambda) = K(\Lambda_{p(x)})$ is a Galois extension over K with $\mathrm{Gal}(K(\Lambda_{p(x)})/K) \simeq (\mathbb{F}_q[x]/p(x))^*$, where $(\mathbb{F}_q[x]/p(x))^*$ is the unit group of the ring $\mathbb{F}_q[x]/(p(x))$. We use σ_A to denote the automorphism of $\mathrm{Aut}(K(\lambda)/K)$ corresponding to A, i.e., $\lambda^{\sigma_A} = \lambda^A$. The size of $(\mathbb{F}_q[x]/p(x))^*$ is denoted by $\Phi(p(x))$. If $p(x)$ is an irreducible polynomial of degree d over $\mathbb{F}_q$, we have $\Phi(p(x)) = q^d - 1$. In this case, the extension $K(\Lambda_{p(x)})/K$ is cyclic and $\mathrm{Gal}(K(\Lambda_{p(x)})/K) \simeq (\mathbb{F}_q[x]/p(x))^* \simeq \mathbb{F}_{q^d}^*$.

3 Construction of subspace design

Let $\sigma \in \mathrm{Aut}(F/\mathbb{F}_q)$ be an automorphism of a finite order. Denote by F^σ the fixed field by $\langle\sigma\rangle$, i.e., $F^\sigma = \{x \in F : x^\sigma = x\}$. By the Galois theory, F/F^σ is a Galois extension and $\mathrm{Gal}(F/F^\sigma) = \langle\sigma\rangle$. Let D be a divisor of F such that $D^\sigma = D$. Assume that Q' is a place of F lying above a rational place Q of F^σ and $Q' \notin \mathrm{supp}(D)$. Furthermore, assume that $\mathcal{V}$ is an $\mathbb{F}_q$-subspace of $\mathcal{L}(D)$ such that $\mathcal{V} \cap \mathcal{L}(D - Q') = \{0\}$.

For each place $P \in \mathbb{P}_F$ such that $P \notin \mathrm{supp}(D)$ and $P, P^{\sigma^{-1}}, \ldots, P^{\sigma^{-(t-1)}}$ are distinct, we define the subspace $\mathcal{H}_P$:

$$\mathcal{H}_P = \{f \in \mathcal{V} : f(P^{\sigma^{-i}}) = 0 \text{ for each } i \in \{0, \ldots, t-1\}\} = \mathcal{V} \cap \mathcal{L}\left(D - \sum_{i=0}^{t-1} P^{\sigma^{-i}}\right). \tag{4}$$

Recall that $f(P)$ is defined to be the residue class of f in the residue field O_P/P. Hence, it is clear that

$$\dim_{\mathbb{F}_q}(\mathcal{H}_P) \geqslant \dim_{\mathbb{F}_q}(\mathcal{V}) + \dim_{\mathbb{F}_q}\left(D - \sum_{i=0}^{t-1} P^{\sigma^{-i}}\right) - \dim_{\mathbb{F}_q}(D) \geqslant \dim_{\mathbb{F}_q}(\mathcal{V}) - t\deg(P).$$

Let $f(P)^\sigma = f(P)^{q^e}$ for some integer $e \geqslant 0$. Thus, we have $f^{\sigma^i}(P^{\sigma^i}) = f(P)^{\sigma^i} = f(P)^{q^{ei}}$ for all integers $i \geqslant 0$.

Define $S_P = \{P^{\sigma^{-i}} : i \in \{0, \ldots, t-1\}\}$, and denote by $\mathcal{F}_r$ a set of places P with degree r such that S_P are disjoint and $|S_P| = t$.

▶ **Theorem 6.** *For any integers s, t with $1 \leqslant s \leqslant t$, the collection $(\mathcal{H}_P)_{P \in \mathcal{F}_r}$ of subspaces of $\mathcal{V}$, each of codimension at most rt, is an $\left(s, \frac{\ell s}{r(t-s+1)}\right)$ strong subspace design, where $\ell = \deg(D)$.*

In the above construction, there is no upper bound on the degree of the divisor D. This makes it possible to compute the dimension of the Riemann-Roch space $\mathcal{L}(D)$. The next construction is for the case when the degree of D is upper bounded by $[F : F^\sigma]$. This construction works for function fields of small genus.

Suppose that there exists a rational place Q in F^σ such that there is only one place Q' of F lying above Q. Let D be a positive divisor of F with $Q' \notin \mathrm{supp}(D)$ and $\deg(D) < n$. For each place $P \in \mathbb{P}_F$ such that $P \notin \mathrm{supp}(D)$ and $P, P^{\sigma^{-1}}, \ldots, P^{\sigma^{-(t-1)}}$ are distinct, we define the subspace $\mathcal{I}_P$:

$$\mathcal{I}_P = \{f \in \mathcal{L}(D) : f(P^{\sigma^{-i}}) = 0 \text{ for each } i \in \{0, \ldots, t-1\}\}. \tag{5}$$

▶ **Theorem 7.** *For any integers s, t with $1 \leqslant s \leqslant t$, the collection $(\mathcal{I}_P)_{P \in \mathcal{F}_r}$ of subspaces of $\mathcal{L}(D)$, each of codimension at most rt, is an $\left(s, \frac{\ell s}{r(t-s+1)}\right)$ strong subspace design, where $\ell = \deg(D)$.*

4 Subspace design from cyclotomic function fields

In this section, we will present subspace design from the construction given in Section 3 by applying cyclotomic function fields. We start with the subspace design in an ambient space of smaller dimension.

The small dimension case. If $\deg(D)$ is smaller than $n = [F : F^\sigma]$ and n is smaller than the genus $\mathfrak{g}(F)$ of F, in general it is hard to compute dimension of the Riemann-Roch space $\mathcal{L}(D)$. Therefore, we cannot use the construction given in Theorem 7. In this subsection, we apply Theorem 7 to the case where we can estimate the dimension of $\mathcal{L}(D)$.

Let F be the rational function field $\mathbb{F}_q(x)$. Let $\sigma \in \mathrm{Aut}(F/\mathbb{F}_q)$ be given by $x \mapsto \gamma x$, where γ is a primitive element of $\mathbb{F}_q^*$. By Theorem 7, one can obtain the subspace design given in [7]. Below we show that the subspace design given in [7] can be realized by using cyclotomic function fields.

Put $K = \mathbb{F}_q(x)$. Let $p_1(x)$ be a monic linear polynomial. For instance, we can simply take $p_1(x) = x$. Then the cyclotomic function field $F_1 := K(\Lambda_{p_1})$ is a cyclic extension over K with $\mathrm{Gal}(F_1/K) \simeq \mathbb{F}_q^*$. In fact, $F_1 = K(\lambda) = \mathbb{F}_q(\lambda)$ with λ satisfying $\lambda^{q-1} + x = 0$. Thus, $K = \mathbb{F}_q(\lambda^{q-1})$. Let γ be a primitive root of $\mathbb{F}_q$ and let $\sigma \in \mathrm{Gal}(F/K)$ be defined by $\lambda^\sigma = \lambda^\gamma = \gamma\lambda$. This gives the exactly the same function fields and automorphism σ as in [7]. Therefore, we conclude that this cyclotomic function field also realizes the subspace design given in [7].

Next we consider a monic *primitive* quadratic polynomial $p_2(x) = x^2 + \alpha x + \beta$ with $\alpha, \beta \in \mathbb{F}_q$. Then the cyclotomic function field $F_2 := K(\Lambda_{p_2})$ is a cyclic extension over K with $\mathrm{Gal}(F_2/K) \simeq (\mathbb{F}_q[x]/(p_2)^*$. In fact, $F_2 = K(\lambda)$ with λ satisfying $\lambda^{q^2-1} + \lambda^{q-1}(x^q + x + \alpha) + x^2 + \alpha x + \beta = 0$. (see [14]). Let σ be a generator of $\mathrm{Gal}(F_2/K)$. Then by the Galois theory, the fixed field F_2^σ is the rational function field $K = \mathbb{F}_q(x)$. The genus of the function field F_2 is $\mathfrak{g}(F_2) = \frac{(q-2)(q+1)}{2}$ [13, 14].

The zero of $p_2(x)$ is the unique ramified place in $\mathbb{F}_q(x)$ and it is totally ramified. Let P' be the unique place of F_2 that lies over the zero of $p_2(x)$. Let ℓ be an even positive integer with $\ell < q^2 - 1$ and let $D = (\ell/2)P'$. Then $\deg(D) = \ell$ and $D^\sigma = D$. Furthermore, we know

that the the zero of $(x-\alpha)$ is fully inert in F_2/K. By Theorem 7, we have the following result.

▶ **Theorem 8.** *For all positive integers s, r, t, m and prime powers q satisfying $s \le t \le m = \zeta q^2$ for some $\zeta \in (0, 1/2]$, the above construction yields a collection of $M = \Omega(\frac{q^r}{rt})$ spaces $\mathcal{I}_1, \ldots, \mathcal{I}_M \subset \mathbb{F}_q^m$, each of codimension rt, which forms an $\left(s', \frac{(1+1/(2\zeta))ms'}{r(t-s'+1)}\right)$ strong subspace design for all $s' \leqslant s$.*

Proof. Choose ℓ such that the dimension of $\mathcal{L}((\ell/2)P')$ is $m = \zeta q^2$. By the Riemman-Roch Theorem, we have $\zeta q^2 \geqslant \deg((\ell/2)P') - \mathfrak{g}(F_2) + 1$, i.e., $\ell \le \zeta q^2 + g - 1 \le (1/2 + \zeta)q^2$. The desired result follows from Theorem 7. ◀

The large dimension case. In this subsection, we will make use of Theorem 6 due to large genus. Let $p(x) \in \mathbb{F}_q[x]$ be a monic primitive polynomial of degree $d \geqslant 2$. Consider the cyclotomic function field $F := K(\Lambda_{p(x)})$, where K is the rational function field $\mathbb{F}_q(x)$. Then F/K is a Galois extension with $\mathrm{Gal}(F/K) \simeq (\mathbb{F}_q[x]/(p(x)))^*$. Thus, $\mathrm{Gal}(F/K)$ is a cyclic group of order $q^d - 1$. Let σ be a generator of this group. Then by the Galois theory, the fixed field F^σ is the rational function field $\mathbb{F}_q(x)$.

The zero of $p(x)$ is the unique ramified place in $\mathbb{F}_q(x)$ and it is totally ramified. Let P' be the unique place of F lying over the zero of $p(x)$. Let Q' be the unique place of F that lies over the zero of x. Since Q' is totally inert, we have $\deg(Q') = [F : F^\sigma] = q^d - 1 := m$.

The genus of the function field F is $\mathfrak{g} = \frac{1}{2}\left(d - 2 + \frac{q-2}{q-1}\right)(q^d - 1) + 1$. Put $D = \left\lceil \frac{2\mathfrak{g}+m-1}{d} \right\rceil P'$. Then $\ell = \deg(D) \geqslant 2\mathfrak{g} + m$ and hence, $\dim_{\mathbb{F}_q}(D - Q') = \deg(D - Q') - \mathfrak{g} + 1$. Choose $\mathcal{V} \subseteq \mathcal{L}(D)$ such that $\mathcal{V}$ and $\mathcal{L}(D - Q')$ are a direct sum of $\mathcal{L}(D)$. Thus, we have $\mathcal{V} \cap \mathcal{L}(D - Q') = \{0\}$ and $\dim_{\mathbb{F}_q}(\mathcal{V}) = \dim_{\mathbb{F}_q}(D) - \dim_{\mathbb{F}_q}(D - Q') = q^d - 1 = m$.

By Theorem 6, we have the following.

▶ **Theorem 9.** *For all positive integers s, r, t, d, m and prime powers q satisfying $\gcd(r, m) = 1$ and $s \le t \le m/r = (q^d - 1)/r$, there is an explicit collection of $M = \Omega(\frac{m \cdot q^r}{rt})$ spaces $\mathcal{H}_1, \ldots, \mathcal{H}_M \subset \mathbb{F}_q^m$, each of codimension at most rt, which forms an $(s', \frac{(d-1/(q-1))ms'}{r(t-s'+1)})$-strong subspace design for all $s' \leqslant s$. Furthermore, the subspace design can be constructed in $\mathrm{poly}(q, m, r)$ time.*

Proof. The subspace design property follows from Theorem 6 since $\ell = \deg(D) \leqslant (d - 1/(q-1))m$. The construction of the subspace design mainly involves finding a basis of $\mathcal{V}$ and evaluations of functions at places of degree r which can be computed in $\mathrm{poly}(q, m, r)$. We can enumerate over all degree r irreducible polynomials $R \in \mathbb{F}_q[x]$ by brute-force in $q^{O(r)}$ time. None of these places are ramified, and each of these places R splits completely into m places of degree r, say $\{P^{\sigma^{i-1}} \mid 1 \leqslant i \leqslant m\}$, in F. So we can pick $b = \lfloor \frac{m}{t} \rfloor$ of these places $P, P^{\sigma^t}, \ldots, P^{\sigma^{(b-1)t}}$, and define a particular subspace of co-dimension rt associated with each of them as in (4). ◀

By setting $t \approx 2s$ and $r \approx \lfloor \frac{\varepsilon m}{2s} \rfloor$ in Theorem 9, we obtain the Main Theorem 4.

References

1 Jean Bourgain and Amir Yehudayoff. Expansion in $\mathrm{SL}_2(\mathbb{R})$ and monotone expanders. *Geometric and Functional Analysis*, 23(1):1–41, 2013. doi:10.1007/s00039-012-0200-9.

2 Zeev Dvir and Amir Shpilka. Towards dimension expanders over finite fields. *Combinatorica*, 31(3):305–320, 2011. doi:10.1007/s00493-011-2540-8.

3 Zeev Dvir and Avi Wigderson. Monotone expanders: Constructions and applications. *Theory of Computing*, 6(12):291–308, 2010. doi:10.4086/toc.2010.v006a012.

4 Michael A. Forbes and Venkatesan Guruswami. Dimension expanders via rank condensers. In *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, pages 800–814, 2015. Extended full version available as ECCC Technical Report TR14-162.

5 Venkatesan Guruswami. Cyclotomic function fields, Artin-Frobenius automorphisms, and list error-correction with optimal rate. *Algebra and Number Theory*, 4(4):433–463, 2010. Preliminary version in STOC 2009.

6 Venkatesan Guruswami. Linear-algebraic list decoding of folded Reed-Solomon codes. In *Proceedings of the 26th IEEE Conference on Computational Complexity*, June 2011.

7 Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016. Preliminary version in FOCS 2013. doi:10.1007/s00493-014-3169-1.

8 Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013. doi:10.1109/TIT.2013.2246813.

9 Venkatesan Guruswami, Carol Wang, and Chaoping Xing. Explicit list-decodable rank-metric and subspace codes via subspace designs. *IEEE Trans. Information Theory*, 62(5):2707–2718, 2016. Preliminary versions in STOC 2013 and RANDOM 2014. doi:10.1109/TIT.2016.2544347.

10 Venkatesan Guruswami and Chaoping Xing. Folded codes from function fields and improved optimal rate list decoding. In *Proceedings of the 44th ACM Symposium on Theory of Computing*, pages 339–350, 2012.

11 Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, Algebraic-Geometric, and Gabidulin subcodes up to the Singleton bound. In *Proceedings of the 45th ACM Symposium on Theory of Computing*, pages 843–852, June 2013. Extended version available as ECCC Technical Report TR12-146.

12 Venkatesan Guruswami and Chaoping Xing. Optimal rate algebraic list decoding using narrow ray class fields. *J. Comb. Theory, Ser. A*, 129:160–183, 2015. Preliminary version in SODA 2014 under slightly different title.

13 David R. Hayes. Explicit class field theory for rational function fields. *Trans. Amer. Math. Soc.*, 189:77–91, March 1974.

14 Liming Ma, Chaoping Xing, and Sze Ling Yeo. On automorphism groups of cyclotomic function fields over finite fields. *Journal of Number Theory*, 169:406–419, 2016. doi:10.1016/j.jnt.2016.05.026.

15 Henning Stichtenoth. *Algebraic function fields and codes*. GMT 254, Springer-Verlag, Berlin, 2008.

Bipartite Perfect Matching in Pseudo-Deterministic NC*

Shafi Goldwasser[1] and Ofer Grossman[2]

1 **Massachusetts Institute of Technology, Cambridge, MA, USA**
shafi@theory.csail.mit.edu
2 **Massachusetts Institute of Technology, Cambridge, MA, USA**
ofer@mit.edu

Abstract

We present a pseudo-deterministic NC algorithm for finding perfect matchings in bipartite graphs. Specifically, our algorithm is a randomized parallel algorithm which uses poly(n) processors, poly($\log n$) depth, poly($\log n$) random bits, and outputs for each bipartite input graph a **unique** perfect matching with high probability. That is, on the same graph it returns the same matching for almost all choices of randomness. As an immediate consequence we also find a pseudo-deterministic NC algorithm for constructing a depth first search (DFS) tree. We introduce a method for computing the union of all min-weight perfect matchings of a weighted graph in RNC and a novel set of weight assignments which in combination enable isolating a unique matching in a graph.

We then show a way to use pseudo-deterministic algorithms to reduce the number of random bits used by general randomized algorithms. The main idea is that random bits can be *reused* by successive invocations of pseudo-deterministic randomized algorithms. We use the technique to show an RNC algorithm for constructing a depth first search (DFS) tree using only $O(\log^2 n)$ bits whereas the previous best randomized algorithm used $O(\log^7 n)$, and a new sequential randomized algorithm for the set-maxima problem which uses fewer random bits than the previous state of the art.

Furthermore, we prove that resolving the decision question $NC = RNC$, would imply an NC algorithm for finding a bipartite perfect matching and finding a DFS tree in NC. This is not implied by previous randomized NC search algorithms for finding bipartite perfect matching, but is implied by the existence of a pseudo-deterministic NC search algorithm.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases Parallel Algorithms, Pseudo-determinism, RNC, Perfect Matching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.87

1 Introduction

Perfect Matching: Computing a maximum matching in a graph is a paradigm-setting algorithmic problem whose understanding has paved the way to formulating some of the central themes of theoretical computer science. In particular, Edmonds [6] proposed the definition of tractable polynomial-time solvable problems versus intractable non-polynomial time solvable problems following the study of the graph matching problem versus the graph clique problem.

* Full version available on ECCC as TR15-208, https://eccc.weizmann.ac.il/report/2015/208/.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 87; pp. 87:1–87:13

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

A distinction of importance to our work is between the **decision** version of the perfect matching problem, which asks whether a perfect matching exists, and the **search** version, which asks to return a perfect matching if any exist. Lovász [15] showed that using randomization, determining the decision problem is reducible to testing that certain integer matrices are non-singular[1]. Since the latter can be done in NC, an RNC algorithm for **deciding** if a perfect matching in a graph exists follows. The **search** version was subsequently shown to be in RNC by Karp, Upfal, and Wigderson [14] via a Monte-Carlo algorithm and by Karloff [13] via a Las-Vegas algorithm.

The next breakthrough was the RNC algorithm of Mulmuley, Vazirani, and Vazirani [16]. They assigned random weights to the edges of the graph and proved the elegant *isolation lemma* which states that with high probability such a random assignment induces (isolates) a unique min-weight perfect matching if at least one exists. Subsequently, the unique minimum weight perfect matching can be determined in parallel by assigning each edge to a different processor whose task is to essentially determine if the edge participates in the unique min-weight perfect matching. However, we emphasize that for the same graph and different isolating weight assignments it is highly likely that different perfect matchings will be found.

Quite recently, a significant step forward has been made by Fenner, Gurjar, and Thierauf [7] who showed how to remove randomization but increase the number of processors, for both the decision and the search variants of the perfect matching problem in bipartite graphs. They show a quasi-NC algorithm: that is, a deterministic poly$(\log n)$ time algorithm which uses quasi-polynomially many processors. A main idea of their work was to construct a set of weight assignments and analyze the union of min-weight matchings with respect to these weight assignments. *In their algorithm, the union of min-weight perfect matchings is never explicitly constructed, but is only used in the analysis.* Indeed, it is not known how to deterministically construct the union of min-weight matchings, and our algorithm uses randomness to construct the union of min-weight matchings. Their algorithm to find the matching follows from applying the procedure of Mulmuley et al [16] to the graph with each of the weight functions they construct until an isolating one is found. Furthermore, the weight assignments used by [7] are different from ours. See the discussion in Section 3 after Lemma 8 of its relation to Lemma 2.3 of [7].

Pseudo-determinism: In a different line of work, initiated by Goldwasser and Gat, [8, 5, 11, 12] the class of search problems which can be solved by *pseudo-deterministic* polynomial time algorithms was introduced – these are probabilistic polynomial-time algorithms for search problems that produce a unique output for each given input except with small probability. That is, they return the same output for all but few of the possible random choices. Algorithms that satisfy the aforementioned condition are named pseudo-deterministic (for a formal definition, see Section 2), as they essentially offer the same functionality as deterministic algorithms: from the point of view of a computationally bounded observer, pseudo-deterministic and deterministic algorithms behave the same, since they always output the same answer.

Efficient pseudo-deterministic algorithms have been shown [5, 8, 12, 11] for several search problems for which no efficient deterministic algorithms are known. These problems include number theoretic search problems [12, 8, 5], multi-variate polynomial non-zero findings [8],

[1] The decision problem is equivalent to testing whether the determinant of the Tutte matrix of the graph (or a simplified version of it in the bipartite case) is identically 0.

and several sub-linear algorithms [11]. The latter work of Goldwasser, Goldreich and Ron [11] shows separations between deterministic, randomized and pseudo-deterministic sub-linear algorithms in accordance with the (asymptotic) number of queries they must require.

The larger question of whether the class of pseudo-deterministic polynomial time search problems is strictly contained in the class of probabilistic polynomial time search problems remains open (see the discussion by Goldreich [10] and in [8, 11] and the discussion below). However, the significance of showing a pseudo-deterministic algorithm in lieu of deterministic ones is amply illustrated by the following observation: If $P = BPP$ then any pseudo-deterministic polynomial time algorithm for a search problem implies a polynomial time deterministic algorithm for the problem.[2] In contrast, the randomized versus deterministic complexity of search problems may not be settled by all proofs of $P = BPP$. That is, certain proofs of $P = BPP$ may not generalize to the search setting. A similar situation emerges for the question of randomized versus deterministic parallel complexity of search problems.

We remark that the study of pseudo-deterministic algorithms seems particularly relevant in the parallel and distributed settings: if two parties invoke a pseudo-deterministic algorithm on the same input, they would be guaranteed to obtain the same result with high probability, regardless of the randomness used. In the distributed setting, pseudo-determinism has been used for scheduling algorithms [9].

1.1 Our Results

In this work we initiate the study of pseudo-deterministic algorithms in the context of NC. In particular, in lieu of deterministic NC algorithms for both the decision and search versions of perfect matching in a graph, we ask the following question: Does a pseudo-deterministic NC algorithm exist for the perfect matching search problem? We settle this question affirmatively for bipartite graphs.

We present a pseudo-deterministic NC algorithm for finding perfect matchings in bipartite graphs. Namely, we present a randomized NC algorithm which on input a bipartite graph G outputs a unique (canonical) perfect matching with high probability, if at least one perfect matching exists. All previous RNC algorithms (including [16]) would output different matchings on different executions.

▶ **Theorem 1** (Main Theorem). *There exists a pseudo-deterministic NC algorithm that, given a bipartite graph G, returns a perfect matching of G, or states that none exist. The algorithm uses $O(\log^2(n))$ random bits.*

Aggarwal, Anderson, and Kao [1] present an RNC algorithm for constructing a depth first search tree for directed graphs. Their algorithm's only use of randomization is to solve bipartite min-weight perfect matching as a subroutine. We can adapt our algorithm to find a unique min-weight perfect matching. Hence, our results imply a pseudo-deterministic NC algorithm for computing depth first search (DFS) in general directed graphs.

▶ **Corollary** (DFS). *There exists a pseudo-deterministic NC algorithm that, given a directed graph G, returns a depth first search tree of G.*

In the full version of the paper, we show a general method for using pseudo-determinism to reducing the number of random bits used by general randomized algorithms. The main

[2] This follows from the characterization of Gat and Goldwasser [8] showing that search problems solvable by polynomial time pseudo-deterministic algorithms are exactly the problems solvable by a polynomial time algorithm with access to a BPP oracle for an analogous decision problem.

idea is that random bits used to solve problems pseudo-deterministically can later be reused, thus avoiding the need to sample more random bits. We then show applications of the technique, including the following Theorem:

▶ **Theorem** (DFS With Few Random Bits). *There exists an RNC algorithm that, given a directed graph G, returns a depth first search tree of G using only $O(\log^2(n))$ random bits. Furthermore, the algorithm is pseudo-deterministic.*

Previously, the best known algorithm for computing a DFS in RNC used $O(\log^7(n))$ random bits (and, furthermore, was not pseudo-deterministic).

It is possible to show, as stated below, that the set of problems solvable by NC pseudo-deterministic algorithms are exactly the set of problems solvable by an NC algorithm with an oracle to RNC decision problems. Thus, our main result implies the existence of a deterministic NC algorithm for finding a bipartite perfect matching if $NC = RNC$. Prior works on the perfect matching search problem in bipartite or general graphs do not imply a deterministic NC solution for the perfect matching search problem, even if an NC algorithm were found for the decision version of the problem. More generally, even if $NC = RNC$, it does not generally imply that every *search* problem that is solvable by an RNC algorithm has a deterministic NC solution.

▶ **Lemma** (Pseudo-deterministic NC). *The class of search problems with pseudo-deterministic NC algorithms is the class of search problems solvable by an NC machine given access to an oracle for RNC decision problems.*

Combining the above lemma with Theorem 1, we prove the following:

▶ **Corollary.** *If $NC = RNC$, then there exists a deterministic NC algorithm that given a bipartite graph G, outputs a perfect matching of G (or states that none exists).*

Combining with the reduction of Aggarwal, Anderson, and Kao [1], we also obtain the following:

▶ **Corollary.** *If $NC = RNC$, then given a graph G, there exists an NC algorithm that returns a depth first search tree of G.*

1.2 High Level Ideas of the Algorithm

We now present an overview of the algorithm and proof of Theorem 1.

Let G be the given bipartite graph. At a high level the algorithm will proceed as follows.

1. In deterministic NC we construct a weight assignment w to the edges of G for which the **union graph** of all min-weight perfect matchings with respect to w (i.e., the union of all edges participating in at least one minimum weight matching) is significantly smaller than G.
2. In randomized NC we **construct** the union graph of all min-weight perfect matchings with respect to w.
3. We repeat steps 1 and 2 above, and every so often we contract some edges of the graph, until we arrive at a graph small enough that we can deterministically construct a perfect matching using brute force.
4. We then “uncontract" to arrive at a matching of the original graph.

Note that the only randomized step of the algorithm is step (2): the construction of the union graph of min-weight perfect matchings. Because the union graph of min-weight perfect matchings is unique (i.e., there is only one union of min-weight perfect matchings), this step

of the algorithm is pseudo-deterministic. As all steps in the algorithm are either deterministic or pseudo-deterministic, the resulting algorithm will be in pseudo-deterministic.

Constructing the union of all min-weight perfect matchings of G with respect to w will be an important step in our solution, as it will allow us to prune the graph (removing the edges which participate in no min-weight perfect matching) while maintaining the property that the graph has a perfect matching. We will next show how to use randomization to construct the union.

We remark that it remains an open problem to deterministically compute the union of min-weight perfect matchings in NC. Whereas the *analysis* of the union graph with respect to a particular set of weight assignments plays an important role in the recent quasi-NC result of [7], *they do not explicitly construct it* as part of their decision or search algorithms.

▶ **Lemma** (Union of min-weight perfect matchings). *Let $G(V, E)$ be a bipartite graph with a polynomially bounded weight assignment w to the edges. Let E_1 be the union of all min-weight perfect matchings in G. There exists an RNC algorithm for finding the set E_1.*

The Lemma appears in Section 3 as Lemma 9. We outline the proof below.

We compute the union of min-weight perfect matchings by creating a process, for each edge e_i, whose goal is to determine whether e_i participates in some min-weight perfect matching. To this end, the process creates a new weight assignment w_i which lowers the weight of e_i by a small amount. The new weight assignment is picked so that if e_i is in some w-minimal perfect matching, then e_i must be in all w_i-minimal perfect matchings; whereas if e_i is not in any w-minimal perfect matching, then it must be in none of the w_i-minimal perfect matchings. By finding any (not necessarily unique) w_i-minimal perfect matching (which can be done in RNC using techniques in [16], and is the only randomized step of our algorithm) and checking whether e_i participates in the matching, we can determine whether e_i is in the union of min-weight matchings with respect to w. We can then return the union of all e_i which are in some min-weight matching.

Recall that the goal of constructing the union graph is to reduce the problem to a smaller graph by removing many edges. To apply the above procedure so as to effectively reduce the size of the graph, we deterministically construct a set of weight assignments with the property that constructing the union of all min-weight perfect matchings in G with respect to these assignments (by going through the weight assignments in sequence and removing edges in each iteration) leaves G with many vertices of degree at most 2. We can then contract all vertices of degree at most 2 with their neighbors to get a smaller graph in which we recursively run our algorithm until we remain with only a constant number of vertices. At this point, we can deterministically compute a unique perfect matching in $O(1)$ time. We note that although performing the contraction procedure in NC takes some care, if it is done properly it is easy to extend a perfect matching in the contracted graph to the original graph.

The construction of weight assignments with the above property proceeds as follows. By a theorem in [2], we learn that if the girth (length of the shortest cycle) of G is at least $4 \log n$, then at least $\frac{1}{10}$ of the vertices have degree at most 2. Therefore, if our weight assignments $w_1, \ldots, w_t$ can make all small cycles disappear (when we construct the union of w_1-minimal matchings, then construct the union of w_2-minimal matchings on this new graph, etc., then at the end are left with a graph with no small cycles), we will be able to reduce our problem to a smaller graph by contracting vertices of degree up to 2. It was shown in [7] that for any weight assignment w, every cycle with nonzero circulation (the sum of the weights of the odd edges of a cycle minus the sum of the weights of the even edges of the cycle) disappears

when we look at the union of w-minimal perfect matchings [3]. We thus need to show how to construct a set of weight functions which will ensure that each small (containing fewer than $4 \log n$ vertices) cycle will have nonzero circulation with respect to at least one of the weight functions.

▶ **Lemma** (Non-Zero Circulation for Small Cycles). *Let G be a bipartite graph on n vertices. Then one can construct in NC a set of $O(\log n)$ weight assignments with weights bounded by* $\text{poly}(n)$ *such that every cycle of length up to $4 \log n$ has nonzero circulation for at least one of the weight assignments.*

This Lemma appears in Section 3 as Lemma 8. We remark that the weight assignments used by [7] are different from ours. We point the reader to a discussion following Lemma 8 for its relation to Lemma 2.3 of [7].

To prove this Lemma we first note that if a cycle of length up to $2k = 4 \log n$ has circulation 0, then the sum of the weights of the odd edges equals the sum of the weights of the even edges. That means that there are two subsets of $E(G)$ of size up to k that have the same sum of weights. If we could construct weight functions such that no two sets of size up to k have the same sum of weights with respect to all of the weight functions, we will have proved the Lemma.

The idea of the construction in the Lemma's proof is to have $k + 1$ weight assignments, and let the mth edge have weight

$$w_i(m) = [m^i]_p$$

with respect to the ith weight function, where $[x]_p$ denotes the number between 1 and p which is equal to x modulo p, and where p is an arbitrary prime greater than n^2.

Then, given the sum of the weights (with respect to each of the w_i) of k elements labeled m_1 through m_k, we can retrieve the sums $\sum_{j=1}^{k} m_j^i \pmod p$, for all $1 \leq i \leq k + 1$. Using these sums, we can use Newton's identities to find the minimal polynomial over $\mathbb{F}_p$ with roots $m_1, m_2, \ldots, m_k$, which uniquely determines the set of elements $m_1, m_2, \ldots, m_k$,. Thus, no two distinct subsets of size up to k can have the same sum of weights.

We note that the weights in the Lemma (where $k = 2 \log n$) are of polynomial size.

We now can construct the union of min-weight matchings with respect to w_1 to get a graph G_1. Then, we can construct the union of min-weight matchings in G_1, with respect to w_2, and so on until w_k. When we are done, we have a graph of high girth, so we can contract many vertices of degree up to 2 (recall that a graph of girth greater than $4 \log n$ has at least one tenth of its vertices of degree up to 2). We now have a smaller graph, and we recurse, completing the proof's outline.

1.3 Pseudo-Determinism and Search vs Decision Derandomization

Understanding the role of randomness in computation is one of the main problems in complexity theory. In the context of *decision* problems the separation between P and BPP indeed captures the gap between randomized and deterministic polynomial time algorithms. However, in the context of *search* problems, it does not. Even if we assume $P = BPP$, there may exist search problems solvable by known randomized polynomial time algorithms

[3] We note that in [7] the authors do not construct the union of w-minimal perfect matchings, but only use the union in their analysis of the algorithm. The algorithm in this paper, on the other hand, constructs the union of w-minimal perfect matchings.

which may not succumb to deterministic polynomial time algorithms. In other words, there exist polynomial time search problems whose randomized vs deterministic complexity may not be settled by a proof of $P = BPP$ (it is worth noting that many approaches towards $P = BPP$, such as pseudorandom generators, may generalize to show that search-P = search-BPP, for certain definitions of search-BPP. In particular, any proof of $P = BPP$ strong enough to prove that promise-P = promise-BPP would generalize to the search setting as shown by Goldreich [10]). For example, the problem of generating primes (given 1^n, output a prime with n bits) has an efficient randomized algorithm. However, even under the assumption $P = BPP$, it is not known if there exists a polynomial time deterministic algorithm. A similar situation is the case for the primitive root problem (given a prime p, output a primitive root modulo p). In [8], the authors prove that a problem admits a pseudo-deterministic polynomial time algorithm (i.e., a randomized search algorithm which outputs the same result for all but few random seeds, formally defined in Section 2) if and only if it is polynomial time reducible to a decision problem in BPP. Thus, any pseudo-deterministic algorithm one demonstrates for a search problem would immediately provide a derandomized algorithm for the problem if $P = BPP$. Our understanding of the randomized complexity for search problems in the parallel setting is quite similar to the polynomial time setting. The NC vs RNC question does not capture the full power of randomization in the parallel setting, since resolving the question for decision problems has no direct bearing on the search-NC vs search-RNC question. We remark that the leading derandomization effort in complexity theory, the so called "hardness vs randomness" paradigm, applies to search problems as well. In a nutshell, the tool of the paradigm is to construct pseudo-random generators or hitting-set generators to derandomize. However, other proofs of $NC = RNC$ may have no direct implication about the relationship of NC and RNC in the context of search. The existence of a pseudo-deterministic algorithm for a problem has direct bearing on the derandomization question under the assumption $NC = RNC$. We show in the full version of the paper that if $NC = RNC$, then the set of problems with NC search algorithms equals the set of problems with NC pseudo-deterministic algorithms (i.e., search algorithms in RNC which output the same solution for all but few random seeds). Our argument is similar to the argument shown in [8] regarding the polynomial time setting. Viewed in this light, our main theorem is that if $NC = RNC$, then there exists a deterministic NC algorithm for *finding* a perfect matching in a bipartite graph. While it would be interesting to find general implications about derandomization of search problems under the assumption that decision problems are derandomized, our paper is specifically about the bipartite perfect matching problem.

1.3.1 Organization

In Section 2, we discuss useful definitions and lemmas from prior works. In Section 3, we prove the main lemmas used in the algorithm. In Section 4, we describe the algorithm, and prove its correctness In the full version of the paper, we show a general method for saving random bits using pseudo-determinism, and apply it to save bits in the matching algorithm, as well as in a depth first search algorithm. Furthermore, in the full version we show that if $NC = RNC$, then all pseudo-deterministic NC algorithms can be fully derandomized. In the full version of the paper, we show how to reduce the number of random bits used by our matching algorithm.

2 Background and Preliminaries

We begin with a formal definition of pseudo-deterministic:

▶ **Definition 2** (Pseudo-deterministic)**.** An algorithm A for a relation R is *pseudo-deterministic* if there exists some function s such that A, when executed on input x, outputs $s(x)$ with high probability, and s satisfies $(x, s(x)) \in R$.

To contrast the definition with that of a standard randomized algorithm, we note that a standard randomized algorithm may output a different y on different executions, as long as $(x, y) \in R$.

▶ **Definition 3** (Pseudo-Deterministic NC)**.** We call an algorithm *pseudo-deterministic NC* if it is in RNC, and is pseudo-deterministic.

We now present some lemmas from previous work.

▶ **Lemma 4** (Theorem 2 in [16])**.** *Given a graph G with a weight function $w : E \to \mathbb{Z}$, with polynomially bounded weights, it is possible to construct a w-minimal perfect matching of G in RNC.*

▶ **Definition 5** (Circulation)**.** Let $G(V, E)$ be a graph with weight assignment w. The *circulation* $c_w(C)$ of an even length cycle $C = (v_1, v_2, \ldots, v_k)$ is defined as the alternating sum of the edge weights of C,

$$c_w(C) = |w(v_1, v_2) - w(v_2, v_3) + w(v_3, v_4) - \cdots - w(v_k, v_1)|.$$

Circulation has been used for an NC algorithm for perfect planar bipartite matching [4] and for a quasi-NC algorithm for bipartite matching [7].

▶ **Lemma 6** (Lemma 3.2 in [7])**.** *Let G be a bipartite graph, and let C be a cycle in G. Let w be a weight function such that the cycle C has nonzero circulation. Then the graph G_1 obtained by taking the union of all min-weight perfect matchings on G does not contain the cycle C.*

The proof in [7] relies on the matching polytope. We present a combnatorial proof found by Anup Rao, Amir Shpilka, and Avi Wigderson, based on Hall's Theorem:

Proof. Let G' be the multigraph obtained by taking the disjoint union of all min-weight perfect matchings (i.e., if an edge e appears in k min-weight perfect matchings of G, then G' contains k copies of e).

Suppose that there exists a cycle C of nonzero circulation in G'. Then suppose without loss of generality that the sum of weights of the odd edges of C is larger than the sum of the weights of the even edges. Then we remove a single copy of each odd edge of C from G', and add a single copy of each even edge of C to G'. Call this new graph G''.

We note that G' is a regular graph since it is the disjoint union of matchings, and matchings are regular graphs of degree 1. We also see that every vertex has the same degree in G'' as in G'. Hence, G'' is regular.

We know that every regular bipartite graph is a union of perfect matchings (to prove this, we can induct on the degree. A regular bipartite graph must satisfy Hall's condition. Therefore, it has a perfect matching, which we can remove. We now obtain a new regular graph of lower degree, which by induction must be a union of perfect matchings).

If we let M be the minimal weight of a matching in G, and we suppose G has d min-weight matchings, then the sum of the weights of edges of G' is Md. However, the total weight of all edges in G'' is lower than the total weight of all edges in G'. We know that G'' is regular of degree d, and therefore is a union of d perfect matchings. If we decompose G'' into d perfect

matchings, it is impossible that they all have weight at least M, because G'' had total weight less than Md. Therefore, G'' has a matching of weight less than M, which corresponds to a matching of weight less than M in G. This contradicts the assumption that M is the minimal weight of a matching in G. ◀

The following lemma originates in [2].

▶ **Lemma 7.** *Let H be a graph with girth (length of shortest cycle) $g \geq 4 \log n$. Then H has average degree < 2.5. In particular, at least $\frac{1}{10}$ (a constant fraction) of the vertices have degree at most 2.*

3 Key Lemmas

We present some definitions, as well as key lemmas from previous work, in Section 2.

In [16], a weight assignment is chosen at random such that with high probability there is a unique min-weight perfect matching. Our goal will be to deterministically construct weight assignments with similar properties. Specifically, we will construct weight assignments which give nonzero circulation to small cycles.

▶ **Lemma 8** (Non-Zero Circulation for Small Cycles)**.** *Let G be a bipartite graph on n vertices. Then one can construct in NC a set of $O(\log n)$ weight assignments with weights bounded by $\text{poly}(n)$ such that every cycle of length up to $4 \log n$ has nonzero circulation for at least one of the weight assignments.*

We would like to point out the differences between this Lemma and Lemma 2.3 of [7] (which originates in [3]). At heart, the two Lemmas are quite different, but they seem similar at first glance. Lemma 2.3 of [7] proves that for any number t, one can construct a set of $O(n^2 t)$ weight assignments with weights bounded by $O(n^2 t)$, such that for any set of t cycles, one of the weight assignments gives nonzero circulation to each of the t cycles.

Since the number of length s cycles is at most n^s, their theorem implies a set of $O(n^{s+2})$ weight assignments with weights bounded by $O(n^{s+2})$ (note that this is quasi-polynomial for $s = 4 \log n$, which will be our setting of parameters) such that at least one of the weight assignments gives non-zero circulation to all small cycles.

We can think about the Lemma as an assignment which isolates all small subsets of S. We will later use this Lemma to construct a weight assignment for the graph G.

Proof. Let $S = \{s_1, \ldots, s_m\}$ be the edges of G. Consider the following weight assignments $w_1, w_2, \ldots, w_{2 \log n + 1}$, where we write $w_i(m)$ as shorthand for $w_i(s_m)$:

$$w_i(m) = [m^i]_p,$$

where $[x]_p$ denotes the number between 1 and p which is equal to x modulo p, and where p is an arbitrary prime greater than n^2. We can find such a prime by having n^2 processes each check a different number between n^2 and $2n^2$. Each of these processes initiate $2n^2$ processes which each test divisibility by an integer up to $2n^2$. (Note that this has no implications regarding generating primes in NC since our input is of size n instead of $\log n$).

We will show that there exist no two subsets of size up to $2 \log n$ which have the same sum of weights. Then, in particular, there exists no cycle of length up to $4 \log n$ with circulation 0.

Suppose there exist two distinct subsets of size up to $k = 2 \log n$ with equal sums of weights with respect to each of the w_i. We can add zeroes to both subsets such that the sizes of the

sets are exactly k. Suppose that the sums of the weights of two subsets $A = \{a_1, a_2, \ldots, a_k\}$ and $B = \{b_1, b_2, \ldots, b_k\}$ are the same. This gives us the following equivalences modulo p:

$$\begin{aligned} a_1 + a_2 + \cdots + a_k &\equiv b_1 + b_2 + \cdots + b_k \pmod{p} \\ a_1^2 + a_2^2 + \cdots + a_k^2 &\equiv b_1^2 + b_2^2 + \cdots + b_k^2 \pmod{p} \\ &\cdots \\ a_1^{k+1} + a_2^{k+1} + \cdots + a_k^{k+1} &\equiv b_1^{k+1} + b_2^{k+1} + \cdots + b_k^{k+1} \pmod{p}. \end{aligned}$$

We claim that this implies that $A = B$. We note that if $a_i \equiv b_j$ modulo p, then $a_i = b_j$ because p is larger than n^2 which is larger than the maximal size of a_i or b_j. Therefore, it will suffice to show that the set A and the set B are equivalent in $\mathbb{F}_p$.

Given the sums of the ith powers of the a_j for i between 1 and $k+1$, Newton's identities uniquely determine the values of the fundamental symmetric polynomials in the a_j. Therefore, Newton's identities also uniquely determine the minimal polynomial which has as roots all of the a_j (with multiplicity). We know that this polynomial will be of degree k and therefore since the b_j share this polynomial, the set of the a_i and the set of the b_j must be equal (they are both the set of roots of the same polynomial), completing the proof that the weight assignment has no two distinct subsets of size up to k with the same sum of weights with respect to all of the weight assignments. ◀

The following lemma shows that in RNC we can construct the union of min-weight perfect matchings of a graph G with a weight assignment w.

▶ **Lemma 9** (Union of min-weight perfect matchings). *Let $G(V, E)$ be a bipartite graph with a polynomially bounded weight assignment w to the edges. Let E_1 be the union of all min-weight perfect matchings in G. There exists an RNC algorithm for finding the set E_1.*

The idea behind the proof is that for each edge e_i, we run a process whose goal is to tell whether e_i is part of a min-weight perfect matching. To do so, the process creates a new weight function which lowers the weight of e_i so that if e_i was in a min-weight perfect matching, under the new weight assignment e_i is in *every* min-weight perfect matching (but if e_i was not in any min-weight perfect matching, it should still not be in any min-weight matching). Then, we use Lemma 4 to find a min-weight perfect matching, and we check if e_i is in the matching. e_i will be in the matching if and only if it is part of a min-weight matching with respect to the original weight function w.

Proof. For each edge $e_i \in E$, consider the weight function w_i defined by

$$w_i(e_j) = \begin{cases} 2w(e_j) - 1 & \text{if } i = j, \\ 2w(e_j) & \text{if } i \neq j. \end{cases}$$

Suppose that M is the minimum weight for a matching with respect to w. Then with respect to w_i, the min-weight matching will have weight $2M$ if e_i is in no w-minimal matching. Otherwise, the min-weight matching will have weight $2M - 1$. By finding a w_i-minimal perfect matching (which we can do in RNC by Lemma 4) and checking its weight (or whether e_i participates in the matching), we can determine whether e_i is in a w-minimal matching.

Note that this is highly parallelizable: we can run the above for each edge in parallel. Then, we return the set of all e_i which are part of some w-minimal matching. ◀

4 The Algorithm

We now put everything together to construct an algorithm:

PERFECT-MATCHING(G)

1 Check if G has a matching in RNC using the algorithm of [16].
2 If G does not have a perfect matching, return $\perp$.
3 If $|E(G)| \leq 100$:
4 Find and return a perfect matching of G using brute force.
5 Let $\{w_1, \ldots, w_t\}$ be the weight assignments from Lemma 8 with $t = O(\log n)$.
6 Let $G_0 = G$.
7 For $i = 1, 2, \ldots, t$:
8 Let G_i be the union of w_i-minimal perfect matchings of G_{i-1} (use Lemma 9).
9 Contract vertices of degree up to 2 in G_t to create G' (*see full version of paper for details*).
10 Let $M' =$ PERFECT-MATCHING(G').
11 Extend the matching M' in G' to a matching M in G_t (*see full version of paper for details*).
12 Return M.

We first argue that the algorithm returns a perfect matching with high probability. To do so, we first note that since G_t and G have the same vertices, it is enough to find a perfect matching on G_t. It is therefore enough to show that G' has a perfect matching, and that in step 11 we can extend the perfect matching M' in G' to a perfect matching M in G_t. This requires analyzing the contraction procedure of step 9. The contraction procedure takes some care, but its ideas are non-central to our proof.

The main idea behind the contraction step is that if we contract both edges adjacent to a vertex of degree 2 and find a matching in the new contracted graph, it is easy to turn a perfect matching in the contracted graph to a perfect matching in the original graph. If v is the vertex of degree 2, and its two neighbors are u_1 and u_2, then once we contract the three vertices we can call the new vertex u'. A perfect matching in the contracted graph will have an edge (v', u') adjacent to u'. That edge, in the original graph, must either be of the form (v', u_1) or of the form (v', u_2) (note that it cannot be of the form (u', v), with v being the vertex of degree 2). Suppose without loss of generality that the edge is (v', u_1). Then we can add the edge (v, u_2) to the matching to form a perfect matching M in G_t from the matching M' in G'. Doing this for multiple vertices in parallel leads to some complications which we elaborate on in the full version of the paper.

Note that we can amplify the success probability of step 8 so that the probability of failure is at most $\frac{1}{n}$. Since the step gets executed a total of $O(\log^2 n)$ times ($O(\log n)$ times on each of the $O(\log(n))$ steps of the recursion), by the union bound the probability that step 8 ever fails is at most $\frac{\log^2(n)}{O(n)}$, which can be further amplifies through repetition.

We now argue the algorithm is pseudo-deterministic. We note that randomization is only used in step 8 to construct the union of min-weight matchings. We use the randomization in the following context: given a weight assignment on a graph, construct the union of min-weight perfect matchings of the graph. Since this has a unique correct answer, correctness implies uniqueness. Therefore, our algorithm returns the same output with high probability, and is therefore pseudo-deterministic.

We will now show the algorithm lies in RNC. We note that step 4 takes $O(1)$ time and step 5 is in NC by Lemma 8. The number of iterations of the loop in step 7 is of length $O(\log(n)^2)$, by Lemma 8, and taking the union of min-weight perfect matchings within the loop in step 8 is in RNC by Lemma 9. Note that if G_{i-1} has a perfect matching, then so

does G_i, since G_i is a non-empty union of perfect matchings of G_{i-1}. Therefore, the loop iterations can be performed in RNC.

By Lemma 8 and Lemma 6, we see that after completing the loop, G_t has no cycles of length up to $4 \log n$. By Lemma 7, in step 9 we contract a constant fraction of the vertices, so G' has a constant fraction of the number of vertices of G_t. Therefore, the number of recursive calls of step 10 is $O(\log n)$.

This completes the algorithm's analysis, proving the following theorem:

▶ **Theorem 10.** *There exists a pseudo-deterministic NC algorithm that, given a bipartite graph G on n vertices, returns a perfect matching of G, or states that none exist.*

We note that as a consequence of Theorem 10 and a result appearing in the full version of the paper (the result that if $NC = RNC$, then every pseudo-deterministic NC algorithm can be fully derandomized), if $NC = RNC$ then the bipartite perfect matching search problem can be solved in NC. In the full version, we improve upon Theorem 10 by showing a pseudo-deterministic NC algorithm for bipartite perfect matching which uses only $O(\log^4 n)$ random bits. In the full version of the paper, we further improve upon that by showing a pseudo-deterministic NC algorithm using only $O(\log^2 n)$ random bits.

5 Discussion

We can adapt our algorithm to bipartite maximum matching. Given a bipartite graph G, we add edges such that we have a complete graph, and give weight 1 to each edge of G and weight 0 to each edge not in G. Now, we take the union of max-weight matchings. We know that any matching on this graph will have the same maximal weight (the symmetric difference of two matchings of different weights will contain a cycle of non-zero circulation). We now pseudo-deterministically find a perfect matching in this new graph, and restrict it to G to output a maximum matching.

The above also implies pseudo-deterministic NC algorithms for some network flow problems such as max-flow approximation, which was shown in [17] to be NC-reducible to maximum bipartite matching. In addition, as an immediate corollary we get a pseudo-deterministic algorithm for max flow, where capacities are expressed in unary (this problem was shown to be NC-reducible to bipartite perfect matching in [14])

It remains open to find a pseudo-deterministic NC algorithm for perfect matching in general (non-bipartite) graphs.

In the context of polynomial time pseudo-determinism, there are many fundamental problems with polynomial time randomized algorithms where the existence of pseudo-deterministic polynomial time algorithms remains open. These problems include generating primes (given 1^n, output a prime with n bits); given a prime p and 1^d, finding an irreducible degree d polynomial over $\mathbb{F}_p$; and finding a primitive root modulo p given a prime p and the factorization of $p-1$. We hope for more progress towards finding pseudo-deterministic polynomial time algorithms for these problems.

Acknowledgments. Thanks to Anup Rao for communicating to us his proof with Amir Shpilka and Avi Wigderson of Lemma 6 (also Lemma 3.2 in [7]). We are very grateful to Oded Goldreich for many helpful discussions on this paper and on the connections between the questions of decision versus search derandomization and pseudo-determinism.

References

1. Alok Aggarwal, Richard J. Anderson, and M.-Y. Kao. Parallel depth-first search in general directed graphs. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 297–308. ACM, 1989.
2. Noga Alon, Shlomo Hoory, and Nathan Linial. The Moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002.
3. Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM Journal on Computing*, 24(5):1036–1050, 1995.
4. Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory of Computing Systems*, 47(3):737–757, 2010.
5. Bart de Smit and Hendrik W. Lenstra. Standard models for finite fields. In *Handbook of finite fields, Discrete Mathematics and Its Applications*. CRC Press, Hoboken, NJ, 2013.
6. Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.
7. Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2016, pages 754–763, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897564.
8. Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 18, page 136, 2011.
9. Mohsen Ghaffari. Near-optimal scheduling of distributed algorithms. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 3–12. ACM, 2015.
10. Oded Goldreich. In a world of P= BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 191–232. Springer, 2011.
11. Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 127–138. ACM, 2013.
12. Ofer Grossman. Finding primitive roots pseudo-deterministically. *ECCC*, 23rd December 2015. URL: http://eccc.hpi-web.de/report/2015/207/.
13. Howard J. Karloff. A Las Vegas RNC algorithm for maximum matching. *Combinatorica*, 6(4):387–391, 1986.
14. Richard M Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 22–32. ACM, 1985.
15. László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979.
16. Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.
17. Maria Serna and Paul Spirakis. Tight RNC approximations to max flow. In *STACS 91*, pages 118–126. Springer, 1991.

A Linear Lower Bound for Incrementing a Space-Optimal Integer Representation in the Bit-Probe Model*

Mikhail Raskin

Department of Computer Science, Aarhus University, Aarhus, Denmark
raskin@mccme.ru

Abstract

We present the first linear lower bound for the number of bits required to be accessed in the worst case to increment an integer in an arbitrary space-optimal binary representation. The best previously known lower bound was logarithmic. It is known that a logarithmic number of read bits in the worst case is enough to increment some of the integer representations that use one bit of redundancy, therefore we show an exponential gap between space-optimal and redundant counters.

Our proof is based on considering the increment procedure for a space optimal counter as a permutation and calculating its parity. For every space optimal counter, the permutation must be odd, and implementing an odd permutation requires reading at least half the bits in the worst case. The combination of these two observations explains why the worst-case space-optimal problem is substantially different from both average-case approach with constant expected number of reads and almost space optimal representations with logarithmic number of reads in the worst case.

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

Keywords and phrases binary counter, data structure, integer representation, bit-probe model, lower bound

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.88

1 Introduction

We consider the problem of representing integers in a certain range by binary codes to support efficient increment operations in the bit-probe model. Our main interest is representing integers in the range $0, \ldots, 2^n - 1$ by exactly n bits.

Most computational tasks require storing integers, and in some cases special encodings are better suited to the task. Probably the first encodings used in bit-based memory are the standard positional binary notation and binary coded decimal. We will consider the task of incrementing an integer counter in the bit probe model. Following [5], an increment algorithm is defined as a **decision assignment tree** (DAT), a binary tree where each inner node specifies a bit of the code that has to be read to make a decision, and every leaf node contains a set of changes to the code to perform. In this model the number of bits read by

* The author acknowledges support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation. This work was partially supported by the French National Research Agency (ANR project GraphEn / ANR-15-CE40-0009).

Table 1 A summary of best known results.

Number of bits read to increment	Space-optimal	Single extra bit
Average	$\Theta(1)$ (binary notation)	
Worst-case	$\geqslant \log_2 n$ [4] $\leqslant n-1$ [2] $\geqslant \frac{n}{2}$ **our contribution**	$\geqslant \log_2 n$ [4] $\leqslant \log_2 n + O(1)$ [8]

an increment is the depth of the corresponding leaf, and the number of bits written is the number of changes in the leaf.

The standard binary notation with n bits reads and writes only 2 bits on average, but has to read and write n bits in the worst case. It is also **space-optimal**, i.e. every combination of bits represents a unique integer. Gray codes [6] allow writing only one bit for each increment operation, but they still require reading all n bits. In the article [5] Fredman introduces the notion of DAT and considers codes that can use more bits than necessary but still require writing only a single bit per update. A logarithmic lower bound is proven for the number of bits read in the worst case for such a code. This bound is sometimes cited in connection with the codes that write a constant number of bits per increment; while the bound is correct, the proof in [5] does use the fact that only one bit is written. Fredman also gave a construction of a code such that the increment procedure needs to read only $O(\log n)$ bits in the worst case, but the code length is worse than for the standard notation by a large multiplicative factor. Frandsen, Miltersen and Skyum consider a more general problem of encoding elements of a generic monoid; their article [4] provides a general lower bound for the number of bits read by an increment procedure for integers in the worst case and a construction of a code with an increment procedure that needs to read $\log_2 n + 1$ bits in the worst case. This code needs $\log_2 n$ extra bits. Constructions developed by Bose et al. in [1] and Rahman and Munro [8] require a single extra bit or less and achieve logarithmic number of bits read by the corresponding increment procedures. Rahman and Munro also prove a lower bound of $\Omega(\sqrt{n})$ bits read in the worst case for the special case when increment reads a different subset of bits for every input or at least each subset of bits is read only for a constant number of inputs (this is a strong condition; for example, the standard binary notation reads only the last bit for half of all the inputs). Elmasry and Katajainen provide a code [3] that uses a logarithmic number of extra bits and requires the increment procedure to read only a logarithmic number of bits in the worst case while ensuring efficient implementation on a word-based RAM machine.

For the space-optimal case there is a code [2] that allows reading $n-1$ bits in the worst case and writes no more than 3 bits for each increment operation. This code has been found by brute force search. The best previously known lower bound on the number of bits to read in the worst case given in [4] is logarithmic in n.

In this paper we close the exponential gap and settle the complexity of the problem up to a multiplicative factor of 2 by proving that every representation of integers from 0 to $2^n - 1$ using n bits require the increment operation to read at least $\frac{n}{2}$ bits in the worst case. The proof uses properties of permutations to explain why the space-optimal codes and the redundant codes are qualitatively different from the point of view of the worst-case complexity of the increment operation.

In the next section we give the standard definitions related to permutations and cite their standard properties. Then we define our model of computation. In the section 3 we give an overview of the core ideas and an outline of the proof.

The Section 4 contains the detailed definitions of the constructions and the proofs of their basic properties. The Section 5 contains the combinatorial details and the final part of the proof. The purpose of the Sections 4 and 5 is to remove any remaining uncertainty after the brief presentation of the proof in the Section 3.

2 Preliminaries

2.1 Algebraic preliminaries

In this subsection we recall algebraic notions and the standard theorems about permutations required by our proof. We follow the definitions from [7], but equivalent definitions can be found in many other abstract algebra books. This subsection can be skipped at the reader's discretion.

▶ **Definition 1.** A **group** is a pair $(G, \circ)$ consisting of a set G and a function $\circ : G \times G \to G$, such that the following conditions hold:

- **Associativity:** $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$.
- **Neutral element:** $\exists e \in G : \forall a \in G : e \circ a = a \circ e = a$.
  e is called a **neutral element** of a group
- **Inverse element:** $\forall a \in G : \exists b \in G : a \circ b = b \circ a = e$.
  b is called an **inverse element** of a.

▶ **Theorem 2.** *There can be only one neutral element in a group G. Let e denote the unique neutral element in the group. Also, for every $a \in G$ there can be only one inverse element. Let a^{-1} denote the unique inverse element of a. The inverse element of a composition $a \circ b$ is $b^{-1} \circ a^{-1}$.*

▶ **Definition 3.** The **symmetric group** S_n is the group of all bijections (one-to-one correspondences) from the set $\{1, 2, 3, \ldots, n\}$ to itself. The group operation $\circ$ is the composition of functions. Each bijection in the symmetric group is called a **permutation**. The neutral element in the symmetric group is the **identity permutation** $\sigma(x) = x$. The inverse element of a permutation σ is the inverse function σ^{-1}, also called the **inverse permutation** of σ. A permutation $\sigma \in S_n$ can be denoted by $\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \sigma(1) & \sigma(2) & \sigma(4) & \cdots & \sigma(n) \end{pmatrix}$. In the present paper $\circ$ will always be written explicitly.

▶ **Definition 4.** Suppose we are given $k \leqslant n$ different elements $x_1, \ldots, x_k \in \{1, \ldots, n\}$. A permutation σ given by

$$\sigma(x_1) = x_2, \quad \sigma(x_2) = x_3, \quad \sigma(x_3) = x_4, \quad \ldots, \quad \sigma(x_{k-1}) = x_k, \quad \sigma(x_k) = x_1$$

and $\sigma(x) = x$ if $x \notin \{x_1, \ldots, x_k\}$ is called a k-**cycle**. It is denoted $\sigma = (x_1\, x_2\, \ldots\, x_k)$. A 2-cycle is also called a **transposition**.

▶ **Definition 5.** Let $\sigma \in S_n$ be a permutation. A pair of indices (i, j) where $1 \leqslant i < j \leqslant n$ is called an **inversion** (of the permutation σ) if $\sigma(i) > \sigma(j)$ (i.e. if the permutation inverts the order in which i and j go).

▶ **Definition 6.** A permutation is called **even** if it has an even number of inversions; otherwise it is called **odd**.

▶ **Theorem 7.** *The composition of two even permutations or two odd permutations is an even permutation. The composition of an even permutation and an odd permutation in any order is an odd permutation. A k-cycle is an odd permutation if k is even and an even permutation if k is odd. The inverse of a permutation has the same parity.*

2.2 The model

▶ **Definition 8.** A **space-optimal code** is an encoding function Enc from $\{0, 1, 2, \ldots, 2^n - 1\}$ to the set of bit sequences of length n. Each code implicitly defines the decoding function $Dec = Enc^{-1}$ and the **increment function** $Inc(x) = Enc(Dec(x) + 1)$. We will also call such codes **counters**.

▶ **Definition 9.** A **decision assignment tree** (DAT) is a binary tree where each inner node specifies a single position in the code and every leaf node contains a set of changes (assignments) in the code. Execution of a DAT on an input bit sequence starts in the root node and then the next node is the left child of the current node if the bit in the specified position of the input code is 0 and the right child of the current node otherwise. When a leaf node is reached, the output is calculated by taking the input and setting the bits in the positions specified for this leaf node to the specified values.

Each DAT defines a function from bit sequences of some length to bit sequences of the same length. We will say that all the nodes visited during execution of DAT on some input (including the root and the leaf node) **handle** this input. The number of bits read is the depth of the corresponding leaf, and the number of bits written to the code is the number of assignments in the leaf.

▶ **Definition 10.** The set $\{0, 1\}^n$ is called an n-dimensional **hypercube**. An element of a hypercube is called a **vertex**. Every vertex has n coordinates. A k-dimensional **face** is a subset of the n-dimensional hypercube defined by specifying the values of some $n - k$ coordinates (we will call these coordinates **fixed**) and allowing all the possible combinations of values of the remaining k coordinates (we will call these coordinates **free**). Each vertex of a hypercube is a bit sequence; we will identify each vertex with the integer it represents in the standard binary notation. The order on the hypercube vertices given by comparing the vertices as integers is called the **lexicographic order**.

3 The bound and the proof outline

The main result of the present paper is: the increment function for every space-optimal code representing integers from 0 to $2^n - 1$ must read at least $\frac{n}{2}$ bits in the worst case. In other words, there is no space optimal code such that the corresponding increment function never reads more than $L(n) := \frac{n}{2} - 1$ bits.

In this section we present an informal outline of the proof. The core idea of the proof is representing the permutation specified by Inc as a composition of two permutations, *Before* and *After*, defined in terms of vertices handled by the same leaf node in the DAT implementing Inc.

Assume that for some n there is a way to encode integers such that the corresponding increment function Inc can be implemented by a DAT that reads at most $L(n)$ bits in the worst case. Without loss of generality we can consider an implementation that always reads exactly $L(n)$ bits. The increment function maps the n-dimensional hypercube into itself and can be considered as a permutation. This permutation is a cycle of length 2^n, and, therefore, an odd permutation.

We will use a representation of the increment function as a composition of two permutations, *Before* and *After*. Each leaf of the DAT implementing Inc handles some $(n - L(n))$-dimensional face of the n-dimensional hypercube. By definition, the restriction of Inc on each of these faces changes some of the bits in the same way for all the vertices in the face.

We also know that Inc is a bijection, therefore only the fixed bits can be changed. This can be interpreted as a parallel translation of the face. The image of each of the faces in F under Inc is again a $(n - L(n))$-dimensional face. Let F denote the set of all the faces handled by any leaf of the DAT. The set of their images, $Inc(F)$, is also a set of faces. Every vertex lies in exactly one face from F and in exactly one face from $Inc(F)$. Let's fix some order of enumeration of F, i.e. $F = \{F_0, F_1, \ldots, F_{2^{n-L(n)}-1}\}$. This also defines an order on $Inc(F)$, namely, $Inc(F) = \{Inc(F_0), \ldots, Inc(F_{2^{n-L(n)}-1})\}$. We can consider three orders on the hypercube: the standard lexicographic ordering; the ordering where we compare two vertices by first compare their corresponding faces in F and fall back to lexicographic order inside each face; and the ordering where we first compare the containing faces from $Inc(F)$. We can enumerate all the vertices of the hypercube according to these three orders. Enumeration in the lexicographic order is the identity permutations. The remaining two orders give non-trivial permutations; we will call them *Before* and *After*. Note that $Inc = After \circ Before^{-1}$, because the i-th vertex in the j-th face of F is by definition of F mapped to the i-th vertex of the j-th face of $Inc(F)$.

We prove that *Before* and *After* are both even. The proof is the same for both permutations. We need to calculate the number of inversions. Every face is enumerated in lexicographic order, so there can be no inversion including two vertices from the same face. When we consider two different faces, they do not intersect and therefore have to have a coordinate which is fixed in both the faces and has a different value. There have to be at least two common free coordinates between the faces if each of them has $L(n) < \frac{n}{2}$ fixed bits and at least one fixed bit position is shared. Faces with two common free coordinates have an even number of inversions using one vertex from each of the faces, because inversions where the less significant common free coordinate doesn't affect the comparison come in multiples of four (two bits in the less significant common position can be flipped at will), and those where the bits in the less significant common position matter come in the multiples of two (the coordinates in the more significant common position must match, and it doesn't matter if both are equal to zero or to one). Therefore the total number of inversions in the permutation *Before* (the same holds for the permutation *After*) is even.

But if *Before* and *After* are both even, $Inc = After \circ Before^{-1}$ has to be even. The contradiction proves that our initial assumption was wrong and the implementation of Inc has to read at least $\frac{n}{2}$ bits in the worst case.

4 Defining the constructions used in the proof

To illustrate some of the notation, we will use the space-optimal integer representation from [2]. This representation was initially found by a brute-force search. Its increment function was presented as a decision assignment tree (Figure 1).

4.1 The cycle defined by the increment function

▶ **Observation 11.** *If the increment function can be described by a decision assignment tree (DAT), it can be represented by a DAT of the same depth with all the leaves having the same depth.*

▶ **Lemma 12.** *For a space-optimal representation of integers in the range $\{0, \ldots, 2^n - 1\}$ the increment function is a bijection of the set of bit strings of length n. By interpreting these strings as integers using standard binary notation, we can represent the increment function as a permutation of $\{0, 1, 2, \ldots, 2^n - 1\}$. This permutation is a cycle of length 2^n. This cycle is an odd permutation.*

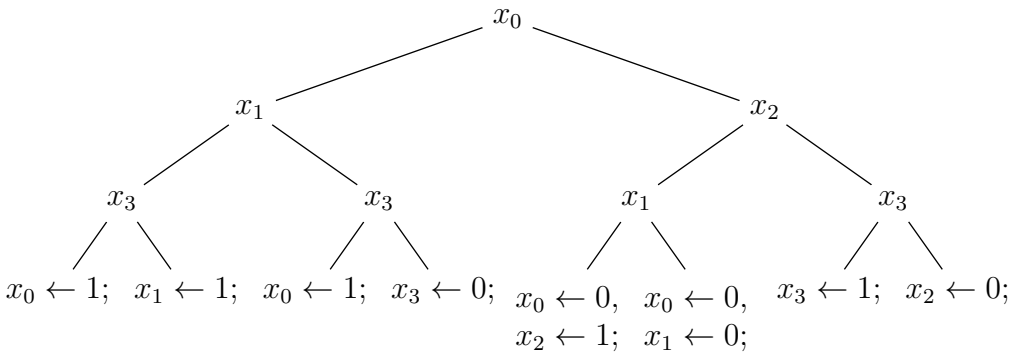


Figure 1 The decision tree from [2].

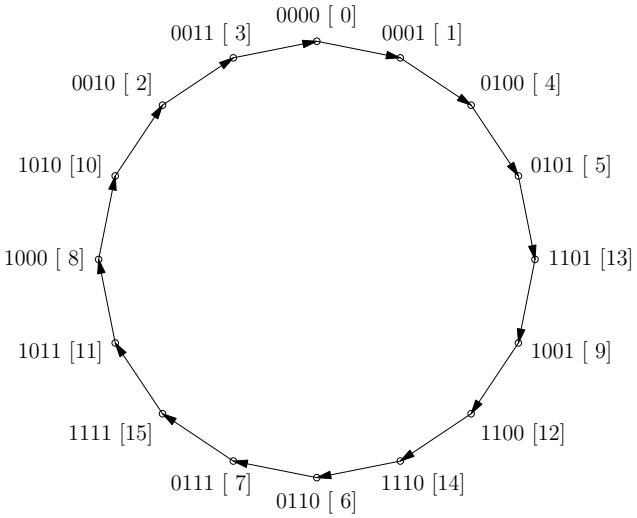


Figure 2 The cycle corresponding to the example from [2].

Proof. The increment function *Inc* maps $Enc(2^n - 1)$ to $Enc(0)$, $Enc(0)$ to $Enc(1)$, $Enc(1)$ to $Enc(2)$, etc. $Enc(0), \ldots, Enc(2^n - 1)$ are all the different binary strings of length n with each string used exactly once, so *Inc* is a bijection. If we interpret the bit strings as integers we get the cycle $(\, Enc(0)\, Enc(1)\, \ldots\, Enc(2^n - 1)\,)$. The cycle is an odd permutation because its length is even.

In the example from [2] the cycle is as shown in the figure. The "first" (x_0) bit from the algorithm's explanation is used as the least significant bit. We can also write this cycle as a table of *Inc* function values:

code	*Inc*(code)	code	*Inc*(code)	code	*Inc*(code)	code	*Inc*(code)
0000 [0]	0001 [1]	0100 [4]	0101 [5]	1000 [8]	1010 [10]	1100 [12]	1110 [14]
0001 [1]	0100 [4]	0101 [5]	1101 [13]	1001 [9]	1100 [12]	1101 [13]	1001 [9]
0010 [2]	0011 [3]	0110 [6]	0111 [7]	1010 [10]	0010 [2]	1110 [14]	0110 [6]
0011 [3]	0000 [0]	0111 [7]	1111 [15]	1011 [11]	1000 [8]	1111 [15]	1011 [11]

The standard notation for this permutation is $\left(\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 1 & 4 & 3 & 0 & 5 & 13 & 7 & 15 & 10 & 12 & 2 & 8 & 14 & 9 & 6 & 11 \end{smallmatrix}\right)$, the cycle notation is (0 1 4 5 13 9 12 14 6 7 15 11 8 10 2 3). This permutation has 39 inversions, so it is an odd permutation. ◀

4.2 Decision assignment trees and the corresponding faces

▶ **Lemma 13.** *Given a DAT for a n-bit integer representation and a node of depth k, the set of all inputs handled (in the sense of Definition 9) by the chosen node is an $(n-k)$-dimensional face.*

Proof. The proof goes by induction. The root has depth 0 and handles all the hypercube, which can be considered an n-dimensional face. A child of node of depth k has depth $k+1$; the set of vertices handled by the child node can be obtained from the set of vertices handled by the parent node by fixing the coordinate inspected in the parent node to one of the two possible values. This coordinate was a free coordinate, so we get an $(n-k-1)$-dimensional face out of a $(n-k)$-dimensional one. ◀

▶ **Lemma 14.** *If a DAT implements a bijection, every coordinate in every assignment in the leaf nodes is a fixed coordinate of the face handled by the corresponding node, i.e. this coordinate is inspected in one of the ancestor nodes of the leaf node.*

Proof. All the vertices need to have different images. Assume a leaf node handled two vertices which differ in the assigned coordinate. This leaf node would handle some face containing both vertices, and this face would also contain some two vertices that differ *only* in the assigned coordinate. But these two latter vertices would have the same image, and this is not allowed. ◀

▶ **Lemma 15.** *If a DAT implements a bijection, the image of the face handled by a leaf node is a translation of this face. In particular, the image is also a face of the hypercube.*

Proof. Changing some of the fixed coordinates of a face performs a parallel translation. ◀

We will now illustrate how the faces are moved. The 4-bit counter using 3 reads for every increment corresponds to a 4-dimensional hypercube. It is more convenient to draw it as two 3-dimensional cubes side by side (so the extra coordinate is projected to the vector proportionate to the projection of the first coordinate). In this case the faces corresponding to the decision assignment tree (DAT) leaves are 1-dimensional faces. We will represent the faces corresponding to the DAT leaves by solid lines. Both pictures use the same set of arrows to represent the movement of the faces. The top picture shows the faces before the moves, and the bottom picture shows the faces after the moves.

We can list the vertices by face. Initially they are split in the following way:

face	vertices	vertices (decimal)	face	vertices	vertices (decimal)
a	0000 and 0100	0 and 4	e	0001 and 1001	1 and 9
b	1000 and 1100	8 and 12	f	0011 and 1011	3 and 11
c	0010 and 0110	2 and 6	g	0101 and 0111	5 and 7
d	1010 and 1110	10 and 14	h	1101 and 1111	13 and 15

and after the faces are moved we get a new split:

face	vertices	vertices (decimal)	face	vertices	vertices (decimal)
a	0001 and 0101	1 and 5	e	0100 and 1100	4 and 12
b	1010 and 1110	10 and 14	f	0000 and 1000	0 and 8
c	0011 and 0111	3 and 7	g	1101 and 1111	13 and 15
d	0010 and 0110	2 and 6	h	1001 and 1011	9 and 11

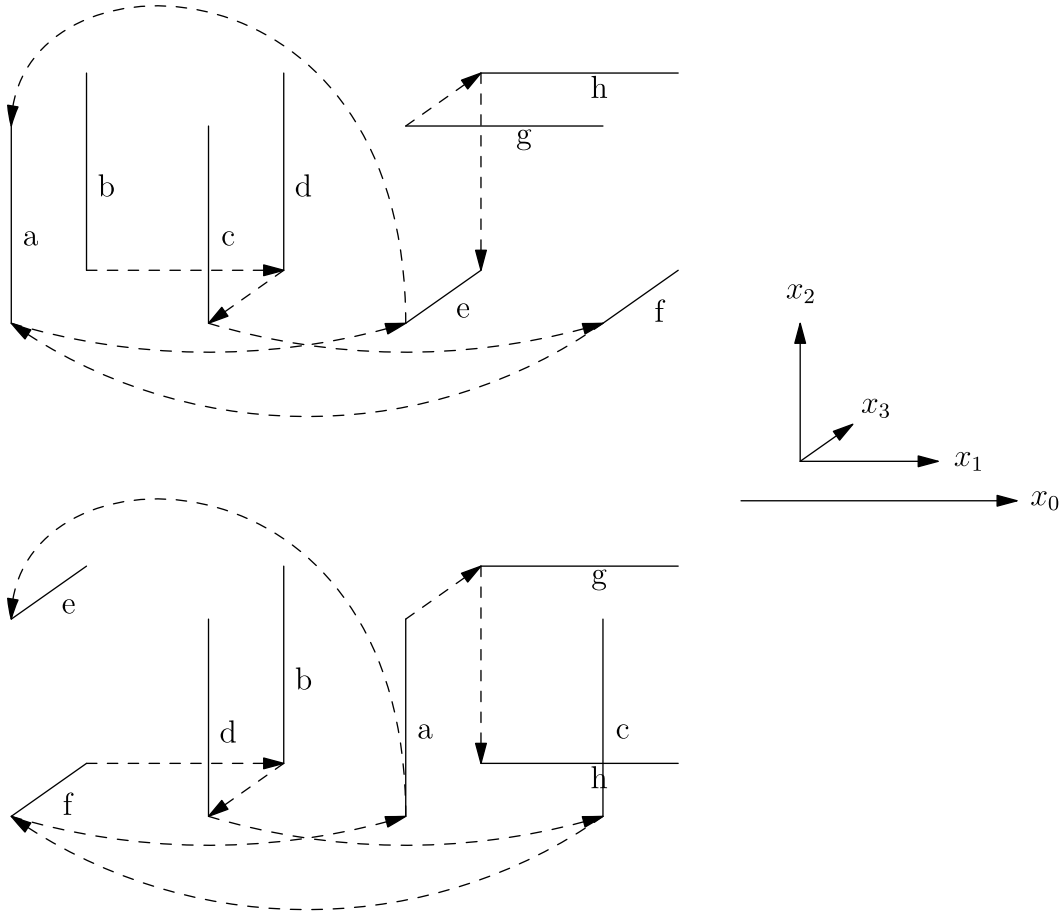


Figure 3 The translations corresponding to the counter from [2]; two pictures show the positions of the 1-dimensional "faces" before and after applying the increment operation.

4.3 Enumerating the faces and the vertices

Assume we have a balanced DAT implementing the increment function *Inc* for a space-optimal integer representation. Each leaf handles the vertices forming a face, these faces are disjoint, have the same dimension and cover the entire hypercube. The *Inc*-images of these faces are again disjoint faces of the same dimension covering the entire hypercube. We need to choose some order on the faces handled by different leaves; it is not important which order we use so we will use the order of leaves in the DAT.

▶ **Definition 16.** Let F_i denote the i-th face in the chosen order.

▶ **Definition 17.** Let *Inc* be a permutation implemented by a balanced DAT of depth l. The *Before* permutation is the enumeration of all the vertices in the hypercube by first enumerating all the vertices in F_0 in the lexicographic order, then all the vertices in F_1, etc. In general, the j-th vertex in the lexicographic order on the face F_i will have the number $i \times 2^l + j$. We could write $Before(i \times 2^l + j) = F_i[j]$. The *After* permutation is defined in a similar way with the j-th vertex in the face $Inc(F_j)$ having the number $i \times 2^l + j$.

▶ **Lemma 18.** *The Inc permutation is the composition of permutations* $Before^{-1}$ *and After, i.e.* $Inc = After \circ Before^{-1}$. *This can also be written as* $\forall k \in \{0, \ldots, 2^n - 1\} : Inc(Before(k)) = After(k)$.

Proof. Let k be represented as $i \times 2^l + j$. We have $Inc(Before(k)) = Inc(Before(i \times 2^l + j)) = Inc(F_i[j]) = Inc(F_i)[j] = After(i \times 2^l + j) = After(k)$ (the j-th element in the i-th face gets translated together with the entire face).

For the example algorithm the $Before(\cdot)$ numbering is:

$$\begin{pmatrix} 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111 \\ 0000\ 0100\ 1000\ 1100\ 0010\ 0110\ 1010\ 1110\ 0001\ 1001\ 0011\ 1011\ 0101\ 0111\ 1101\ 1111 \end{pmatrix}$$

in binary, or $\begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15 \\ 0\,4\,8\,12\,2\,6\,10\,14\,1\,9\,3\,11\,5\,7\,13\,15 \end{pmatrix}$ in decimal notation. The $After$ numbering is

$$\begin{pmatrix} 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111 \\ 0001\ 0101\ 1010\ 1110\ 0011\ 0111\ 0010\ 0110\ 0100\ 1100\ 0000\ 1000\ 1101\ 1111\ 1001\ 1011 \end{pmatrix}$$

or $\begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15 \\ 1\,5\,10\,14\,3\,7\,2\,6\,4\,12\,0\,8\,13\,15\,9\,11 \end{pmatrix}$. We first enumerate the two vertices in the a face, then the two vertices in the b face, etc. using the positions of the faces before and after the move, respectively. We can see that the $Before$ permutation is odd and the $After$ permutation is even. As the example algorithm reads more than a half of all the bits, getting an odd permutation is possible.

As an illustration, 10 is the first vertex in the d face before the translation of the face by the increment function, $Inc(10) = 2$. The first vertex in the face d (the fourth face) has number 6 in the $Before$ ordering; we see that $Before^{-1}(10) = 6$. After the shift the first vertex in the face d is 2. We see that $After(6) = 2$ and $Inc(10) = After(Before^{-1}(10)) = (After \circ Before^{-1})(10) = 2$. ◀

5 Calculating the parity of the permutations

Both of the permutations $Before$ and $After$ are specified in the same way, by cutting the hypercube into faces and enumerating the faces. It is now sufficient to show that any permutation specified in that way is even if the faces have no more than $L(n) = \frac{n}{2} - 1$ fixed coordinates. We will prove that the $After$ permutation is even; exactly the same proof will work for the $Before$ permutation.

5.1 Faces and the inversions

Recall that an inversion of a permutation σ is a pair of numbers $x < y$ such that $\sigma(x) > \sigma(y)$.

▶ **Lemma 19.** *There are no inversions of the permutation After such that $After(x)$ and $After(y)$ are in the same face.*

Proof. If $After(x)$ and $After(y)$ are in the same face and $x < y$ then x has a lower number inside the face than y. But the face is enumerated in the lexicographic order, so $After(x) < After(y)$. ◀

▶ **Lemma 20.** *Consider two faces, $Inc(F_i)$ and $Inc(F_{i'})$ such that $i < i'$. The number of inversions such that $After(x) \in Inc(F_i)$ and $After(y) \in Inc(F_{i'})$ is even.*

▶ Note 21. If $i > i'$ then there are no inversions because x would always be larger than y.

Proof. Note that the vertices are enumerated face-by-face, so the condition that $After(x) \in Inc(F_i)$ and $After(y) \in Inc(F_{i'})$ guarantees $x < y$. Every vertex in each face has exactly one number, so we can just count the number of pairs (u, v) where $u \in Inc(F_i)$ and $v \in Inc(F_{i'})$ and $u > v$. We will use the assumption that each of the faces has $L(n) = \frac{n}{2} - 1$ fixed coordinates. This means that no more than $n - 2$ coordinates are fixed in any of the two

faces. Therefore there are at least two coordinates that are free for both faces. We can write the faces' coordinates one on top of the other one: $\begin{smallmatrix}0&0&*\\0&*&1\end{smallmatrix}\cdots$ Let us consider the least significant of the common free coordinates and call it p.

We will split all the pairs (u, v) into two groups based on the number of coordinates that have to be checked to perform a lexicographic comparison if we read from the most significant bit. Either it is enough to read only some of the coordinates that are more significant than p, or we need to to read the coordinate p and maybe some more.

(1) The number of pairs of codes where the comparison can be made without considering the bits on the position p (and less significant positions) is even, because half of these codes have 0 in the first face on the position p and the other half have 1.

(2) If we have to consider the bits in the position p, the bits in every more significant common position must be equal. There is an even number of such pairs because changing a pair of bits in the same position from $0, 0$ to $1, 1$ doesn't affect the comparison.

We have split all the pairs with $u > v$ into two even-sized sets, as we have to consider either only bits more significant that the position p or the bits including position p. Therefore the total number of such pairs is even.

This finishes the proof that the number of inversions containing two vertices in the two given faces is even. ◀

5.2 Summarizing the inversion counts

▶ **Lemma 22.** *The After permutation (and the Before permutation) for a Inc function represented by a DAT of depth less than $\frac{n}{2}$ are even.*

Proof. In the previous subsection we have proven that every inversion of the *After* permutation has to include elements from different faces (Lemma 19). We have also proven that for every pair of faces the number of inversions represented by their elements is even (Lemma 20). If we sum the inversions for all the pairs of faces we get all the inversions of the permutation. Therefore the number of inversions of the permutation is even. ◀

Now we can prove the main theorem.

▶ **Theorem 23.** *The increment function for every space-optimal binary code representing integers from 0 to $2^n - 1$ must read at least $\frac{n}{2}$ bits in the worst case. In other words, there is no space optimal binary code such that the corresponding increment function never reads more than $L(n) := \frac{n}{2} - 1$ bits.*

Proof. If there is an increment function for a space-optimal binary integer representation reading at most $L(n)$ bits in the worst case, the corresponding *Before* and *After* permutation would both be even. Then the *Inc* function would be an even permutation. But the increment function for a space-optimal binary integer representation has to be a cycle of length 2^n, i.e. an odd permutation. The contradiction proves that our assumption was impossible. ◀

6 Handling a weaker definition of increment

Our definition of increment assumed that incrementing the largest value always yields zero. This requirement can be removed from the definition.

▶ **Theorem 24.** *Consider an increment procedure for a space-optimal integer representation that correctly handles all the possible values except the maximum and always leaves at least two bits unread. Such a procedure always maps the encoding of the maximum value to the encoding of zero.*

Proof. Let n denote the total amount of bits in the code. Let us assume that the increment procedure applied to the maximum values doesn't yield zero. Let k denote a position where the encoding of zero and the result of incrementing the encoding of the maximum value differ. Without loss of generality we can assume that the encoding of zero has 0 at the position k and the value $a = Inc(Enc(2^n - 1))$ has 1 at the position k.

Let us count the vertices x such that $Inc(x)$ has the value 1 at the position k. Almost all vertices have exactly one preimage, $Enc(0)$ has no preimages and a has 2 preimages, so the answer should be $2^{n-1} + 1$. On the other hand, a face corresponding to a vertex of the DAT can have no, all or half of its vertices in the set, depending on the orientation; in any case this is an even number and the total sum has to be an even number.

The contradiction proves that our assumption is false and incrementing the maximum values has to yield zero as the result. ◀

7 Future directions

Minor tweaks of the presented proof allow to extend the result to cover nondeterministic increment procedures. For $n > 10$ the same bound can be proven even if we allow arbitrary changes of the inspected bits together with an arbitrary reversible linear transformation of the unread bits in every leaf node of a DAT.

Closing the gap between the $\frac{n}{2}$ lower bound and $n - 1$ upper bound remains an open problem. Our conjecture is that the true value is $n - o(n)$.

Acknowledgements. I am grateful to Gerth Brodal for attracting attention to this problem and his help with editing the present paper. I am extremely grateful to Gudmund Frandsen for a lot of useful discussions and the efforts he has spent on reading multiple draft versions of this proof. I am grateful to the anonymous reviewers of this and previous versions of the present paper for their valuable advice regarding presentation. I am grateful to an anonymous reviewer for the suggestion that the existence of common fixed coordinates for disjunct faces improves the bound by one; and for the suggestion that the image of the maximum element can be proven to be zero even if this assumption is not included in the definition.

References

1 Prosenjit Bose, Paz Carmi, Dana Jansens, Anil Maheshwari, Pat Morin, and Michiel H. M. Smid. Improved methods for generating quasi-gray codes. In *Algorithm Theory – SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings*, pages 224–235, 2010. doi:10.1007/978-3-642-13731-0_22.

2 Gerth Stølting Brodal, Mark Greve, Vineet Pandey, and Srinivasa Rao Satti. Integer representations towards efficient counting in the bit probe model. *J. Discrete Algorithms*, 26:34–44, 2014. doi:10.1016/j.jda.2013.11.001.

3 Amr Elmasry and Jyrki Katajainen. In-place binary counters. In *Mathematical Foundations of Computer Science 2013 – 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, pages 349–360, 2013. doi:10.1007/978-3-642-40313-2_32.

4 Gudmund Skovbjerg Frandsen, Peter Bro Miltersen, and Sven Skyum. Dynamic word problems. *J. ACM*, 44(2):257–271, 1997. doi:10.1145/256303.256309.

5 Michael L. Fredman. Observations on the complexity of generating quasi-gray codes. *SIAM J. Comput.*, 7(2):134–146, 1978. doi:10.1137/0207012.

6 F. Gray. Pulse code communication, March 17 1953. US Patent 2,632,058. URL: https://www.google.com/patents/US2632058.

7 N. Lauritzen. *Concrete Abstract Algebra: From Numbers to Gröbner Bases*. Concrete Abstract Algebra: From Numbers to Gröbner Bases. Cambridge University Press, 2003.

8 M. Ziaur Rahman and J. Ian Munro. Integer representation and counting in the bit probe model. *Algorithmica*, 56(1):105–127, 2010. doi:10.1007/s00453-008-9247-2.

Rerouting Flows When Links Fail*†

Jannik Matuschke[1], S. Thomas McCormick[2], and Gianpaolo Oriolo[3]

1 **TUM School of Management and Department of Mathematics, Technische Universität München, Munich, Germany**
jannik.matuschke@tum.de
2 **Sauder School of Business, University of British Columbia, Vancouver, Canada**
tom.mccormick@sauder.ubc.ca
3 **Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università di Roma "Tor Vergata", Roma, Italy**
oriolo@disp.uniroma2.it

Abstract

We introduce and investigate *reroutable flows*, a robust version of network flows in which link failures can be mitigated by rerouting the affected flow. Given a capacitated network, a path flow is reroutable if after failure of an arbitrary arc, we can reroute the interrupted flow from the tail of that arc to the sink, without modifying the flow that is not affected by the failure. Similar types of restoration, which are often termed "local", were previously investigated in the context of network design, such as min-cost capacity planning. In this paper, our interest is in computing maximum flows under this robustness assumption. An important new feature of our model, distinguishing it from existing max robust flow models, is that no flow can get lost in the network.

We also study a tightening of reroutable flows, called *strictly reroutable flows*, making more restrictive assumptions on the capacities available for rerouting. For both variants, we devise a reroutable-flow equivalent of an s-t-cut and show that the corresponding max flow/min cut gap is bounded by 2. It turns out that a strictly reroutable flow of maximum value can be found using a compact LP formulation, whereas the problem of finding a maximum reroutable flow is *NP*-hard, even when all capacities are in $\{1, 2\}$. However, the tightening can be used to get a 2-approximation for reroutable flows. This ratio is tight in general networks, but we show that in the case of unit capacities, every reroutable flow can be transformed into a strictly reroutable flow of same value. While it is *NP*-hard to compute a maximal integral flow even for unit capacities, we devise a surprisingly simple combinatorial algorithm that finds a half-integral strictly reroutable flow of value 1, or certifies that no such solutions exits. Finally, we also give a hardness result for the case of multiple arc failures.

1998 ACM Subject Classification G.2.2 [Graph Theory] Graph Algorithms

Keywords and phrases network flows, network interdiction, robust optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.89

1 Introduction

Network infrastructures for transportation, communication, or energy transmission are an important backbone of our society. However, they are also prone to failure or intentional

* A full version of this article is available at https://arxiv.org/abs/1704.07067.
† This work was supported by the Alexander von Humboldt Foundation with funds of the German Federal Ministry of Education and Research (BMBF) and by an NSERC Discovery Grant.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 89; pp. 89:1–89:13


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

sabotage, and in such cases it is desirable to quickly recover the service provided through the network. A crucial frequent requirement of actual network restoration techniques is that restoration is handled locally [13]. As a motivating example, consider a communication network in which data packets are routed along paths. When a link in the network fails, it is desirable to only reroute the traffic that is actually affected by the failure, i.e., those paths that traverse the failing link, without changing or rerouting any part of the flow that is not affected by the failure. Note that arbitrary rearrangement of the flow after a failure is in general more powerful, but it is both undesirable to interrupt customer service and hard to do so reliably and safely [9, 16].

To cope with such a situation, we introduce the concept of reroutable network flows: A flow on s-t-paths is *reroutable* if after failure of any arc $\bar{a} = (\bar{v}, \bar{w})$ in the network, we can reroute all flow that was traversing $\bar{a}$ from $\bar{v}$ to the sink t, while not changing any flow that was not affected by the interruption. Similar concepts were previously discussed in a few other papers [6, 7, 17, 18], but with an emphasis on network design issues, e.g., minimizing the cost of the installed capacity. In contrast, our interest is in computing maximum flows (but we point out that a potential application are feasibility/separation subroutines for capacity reservation). Note that in this setting, we cannot simply send a standard maximum flow, as we need to leave space for rerouting. Before we discuss our findings and better relate them to existing literature, let us formalize the definition of our model.

Network flows. Let $D = (V, A)$ be a digraph with source $s \in V$, a sink $t \in V$ and arc capacities $u \in \mathbb{R}^A_+$. Let $\mathcal{P} \subseteq 2^A$ be the set of simple[1] s-t-paths in D. For arcs $a, \bar{a} \in A$, define

$$\mathcal{P}_a := \{P \in \mathcal{P} : a \in P\} \quad \text{and} \quad \mathcal{P}_{\bar{a} \to a} := \{P \in \mathcal{P} : a, \bar{a} \in P,\, \bar{a} \prec_P a\},$$

where $\bar{a} \prec_P a$ means that P traverses $\bar{a}$ before a. An *s-t-flow* is a vector $x \in \mathbb{R}^{\mathcal{P}}_+$ that assigns a flow value $x(P) \geq 0$ to each $P \in \mathcal{P}$ such that the arc flow values $x(a) := \sum_{P \in \mathcal{P}_a} x(P)$ fulfill the capacity constraint $x(a) \leq u(a)$ for all $a \in A$. The value of a flow x is $\mathrm{val}(x) := \sum_{P \in \mathcal{P}} x(P)$.

Reroutable flows. Let x be an s-t-flow. If an arc $\bar{a} = (\bar{v}, \bar{w}) \in A$ fails, all flow on paths containing the failing arc gets interrupted when it reaches $\bar{v}$. For any $a \in A \setminus \{\bar{a}\}$, we define the *available capacity* of a after failure of $\bar{a}$ by

$$\bar{u}_{x,\bar{a}}(a) := u(a) - \sum_{P \in \mathcal{P}_a \setminus \mathcal{P}_{\bar{a} \to a}} x(P).$$

A *rerouting* of x for the failing arc $\bar{a}$ is a $\bar{v}$-t-flow $x_{\bar{a}}$ of value $x(\bar{a})$ in $(V, A \setminus \{\bar{a}\})$ with capacities $\bar{u}_{x,\bar{a}}$. The flow x is *reroutable* if for every failing arc $\bar{a} \in A$ there is a rerouting $x_{\bar{a}}$ of x.

Strictly reroutable flows. A rerouting $x_{\bar{a}}$ of a flow x for a failing arc $\bar{a}$ is *strict* if $x_{\bar{a}}(a) \leq \bar{u}_x(a) := u(a) - x(a)$ for every $a \in A \setminus \{\bar{a}\}$. We say that x is *strictly reroutable* if for every failing arc $\bar{a} \in A$ there is a strict rerouting of x.

Strictly reroutable flows are both a helpful tool for computing reroutable flows and interesting in their own right, in situations where more conservative assumptions have to be

[1] All our results also work for the case that $\mathcal{P}$ contains non-simple paths, but we restrict to simple paths for ease of notation.

made on the capacities available for rerouting. A natural question is what is the maximum flow value that can be sent by a (strictly) reroutable flow in a given network. We denote the corresponding optimization problem as Max RF and Max SRF, respectively.

1.1 Our results

Complexity of the problems (Sections 2.1 and 2.2). We observe that Max SRF can be solved in polynomial time by formulating it as a linear program. In contrast, Max RF is *NP*-hard, even when $u(a) \in \{1, 2\}$ for all $a \in A$. On the positive side, by showing that the maximum value of a reroutable flow is at most twice as large as the maximum value of a strictly reroutable flow, we obtain a 2-approximation for Max RF for arbitrary capacities. The problem can further be solved exactly in unit capacity networks (see below).

Max flow/min cut gap (Section 2.3). Max flow/min cut results play a central role in network flow theory. We devise a combinatorial upper bound for the maximum reroutable flow value, called *R-cut*, and prove that the corresponding flow/cut gap for both reroutable and strictly reroutable flows is bounded by 2. In fact, our proof is constructive and provides a combinatorial 2-approximation algorithm for the minimum capacity R-cut problem.

Unit capacity networks (Section 3). We consider the case of unit capacities. It turns out that in this case, Max RF and Max SRF are equivalent. Our proof is based on a careful uncrossing argument that allows to transform any reroutable flow into a strictly reroutable flow.

Computing (half-)integral solutions (Section 4). A common property of many flow problems is the existence of an integral optimal solution when capacities are integral. In the case of reroutable flows, this property does not hold. In fact, if we require flow to be integral, the problem becomes *NP*-hard, even for sending a single unit of flow in a unit capacity network. However, for this special case, we devise a simple combinatorial algorithm that computes a half-integral solution or certifies that no flow of value 1 exists. Via our max flow/min cut analysis we also show how to compute 2-approximate half-integral solutions.

Multiple arc failures (Section 5.2). We consider the natural generalization of our problems to multiple simultaneous arc-failures. We show that in this case both variants of the problem are *NP*-hard, even when only two arcs can fail and all arcs have unit capacity. All hardness results in this paper are based on reduction from an intermediary problem, called Forbidden Pairs s-t-Path. They are therefore grouped together in Section 5.

1.2 Related work

As we already pointed out above, "local" rerouting schemes, i.e., schemes that only change flow affected by the failure, have been investigated in network design. A routing scheme in which flow has to be sent along arc-disjoint paths was investigated in [6], see also [18]. The problem of finding a local rerouting from the tail to the head of a failed arc was investigated in [7] and [17]. However, in all these papers the focus was on min-cost capacity planning.

Concepts that deal with the maximization of flow subject to robustness constraints commonly fall under the moniker of *robust flows*. Aggarwal and Orlin [2] studied *k-route flows*. Such a flow is a conic combination of elementary flows, each of which consists of a uniform flow along k disjoint paths. Because of this structure, the failure of any arc can only

destroy a $1/k$ fraction of the flow. A maximum k-route flow can be computed in polynomial time by means of a parametric max flow problem. Another classic model is the *maximum robust flow problem*: Here, the goal is to find a path flow that maximizes the surviving flow after a worst-case failure of k arcs. Aneja et al. [3] showed that for $k = 1$ both an optimal fractional and an optimal integral solution can be found in polynomial time. If k is not bounded by a constant the problem is *NP*-hard [10], but the complexity for any constant value $k \geq 2$ is open. Bertsimas et al. [4] provide an $\Omega(1/k)$-approximation algorithm for the maximum robust flow. Robust flows are closely related to *network flow interdiction*, which takes a dual perspective: The goal is to find a subset of arcs whose removal minimizes the maximum flow value in the remaining network; see the recent article by Chestnut and Zenklusen [8] for an up-to-date overview of this topic.

To the best of our knowledge, the only other flow maximization model that allows for adjustment after the failure are *adaptive flows*, first introduced by Bertsimas et al. [5]: In the first step, an arc flow is specified. After failure of k arcs, a new flow is sent, with the flow value on every arc being bounded by the original flow value. Note that adaptive flows differ from reroutable flows in two important aspects: Adaptive flows allow flow to be 'lost' (the flow value after the failure is lower than the original flow value), whereas in reroutable flows all flow has to reach the sink. Furthermore, adaptive flows can reconfigure the flow in the entire network, whereas in reroutable flows, only the flow affected by the failure can be rerouted.

Another model closely related to reroutable flows is the *online replacement path* problem (ORP) introduced by Adjiashvili et al. [1]. The ORP is a generalization of the shortest path problem: Given a digraph with costs on the arcs, we have to specify an s-t-path. Along the path, we may encounter a failing arc $\bar{a} = \{\bar{v}, \bar{w}\}$, and we have to find a replacement path from $\bar{v}$ to t avoiding $\bar{a}$. The goal is to minimize the total traveled distance, assuming $\bar{a}$ is chosen by an adversary. Adjiashvili et al. [1] show that the ORP can be solved in polynomial time, even when a constant number of arcs fail.

2 LP formulation, approximation, and max flow/min cut

In this section, we discuss the complexity of the two problems and provide bounds on the gap between Max RF and Max SRF. We also introduce an analogue to minimum cuts for reroutable flows and bound the corresponding duality gap. At the end of the section, we show that all our bounds are tight.

2.1 Complexity of Max RF and Max SRF

We now consider an LP formulation for Max SRF. For $\bar{a} \in A$, let $\mathcal{R}(\bar{a})$ be the set of all tail$(\bar{a})$-t-paths in $(V, A \setminus \{\bar{a}\})$, which are exactly the paths that a rerouting for failing arc $\bar{a}$ can use.

$$
\begin{aligned}
[\mathrm{LP}_{\mathrm{strict}}] \qquad \max \quad & \sum_{P \in \mathcal{P}} x(P) & & \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_a} x(P) + \sum_{R \in \mathcal{R}(\bar{a}) \,:\, a \in R} x_{\bar{a}}(R) \;\leq\; u(a) & & \forall\, a, \bar{a} \in A \\
& \sum_{P \in \mathcal{P}_{\bar{a}}} x(P) - \sum_{R \in \mathcal{R}(\bar{a})} x_{\bar{a}}(R) \;=\; 0 & & \forall\, \bar{a} \in A \\
& x, x_{\bar{a}} \geq 0 & & \forall\, \bar{a} \in A
\end{aligned}
$$

The first set of constraints bound the capacities for each rerouting; note in particular that for $\bar{a} = a$, the second term becomes 0, ensuring $x(a) \leq u(a)$ for all $a \in A$. The second set of constraints ensures that the rerouting flow $x_{\bar{a}}$ has value $x(\bar{a})$. Although [$\text{LP}_{\text{strict}}$] has an exponential number of variables, it can be solved in polynomial time via dual separation.

▶ **Theorem 1.** Max SRF *can be solved in polynomial time.*

For the special case of unit capacity networks, we show in Section 3 that an optimal solution to [$\text{LP}_{\text{strict}}$] is also optimal for Max RF.

▶ **Theorem 2.** *For $u \equiv 1$,* Max RF *can be solved in polynomial time.*

An LP for Max RF can be obtained by replacing $\sum_{P \in \mathcal{P}_a} x(P)$ by $\sum_{P \in \mathcal{P}_{\bar{a} \to a}} x(P)$ in the capacity constraints of [$\text{LP}_{\text{strict}}$]. Unfortunately, this modification prevents the dual separation approach from working. In fact, it turns out that Max RF is hard as soon as two different capacities occur. The proof of this result is discussed in Section 5.1.

▶ **Theorem 3.** Max RF *is NP-hard, even when $u(a) \in \{1, 2\}$ for all $a \in A$.*

2.2 Reroutable flows vs. strictly reroutable flows

As Max SRF is a tightening of Max RF, the optimal value of the former is at most that of the latter. We show that the gap between the two values cannot be larger than 2. As we can compute maximum strictly reroutable flows, we obatain a 2-approximation for Max RF.

▶ **Lemma 4.** *Let x be an s-t-flow. If x is strictly reroutable, then x is reroutable. If x is reroutable, then $\frac{1}{2}x$ is strictly reroutable.*

▶ **Corollary 5.** *There is a 2-approximation algorithm for* Max RF.

2.3 Max flow/min cut gap for reroutable flows

An *s-t-cut* is a set of arcs that intersects every s-t-path. Its *capacity* is the sum of capacities of its arcs. A fundamental result in network flow theory is that the value of a maximum s-t-flow is equal to the capacity of a minimum s-t-cut. This result has been successfully generalized to many variants of network flows, such as abstract flows [14] or flows over time [11]. However, in other cases, such as multicommodity flows, the equality does not hold and instead, researchers investigate the worst case ratio between maximum flow and minimum cut; see, e.g., [15].

We present a counterpart to an s-t-cut for reroutable flows. It turns out that max flow and min cut are not necessarily equal and we give a tight bound on the corresponding max flow/min cut gap. An *R-cut* is a set of arcs $R \subseteq A$ together with a collection of cuts $(C_a)_{a \in R}$, where each C_a is a tail(a)-t-cut containing a. We denote $(R, (C_a)_{a \in R})$ by (R, C) for short. The capacity of the R-cut (R, C) is

$$\text{cap}(R, C) := \phi(R, C) + \sum_{a \in R} u(C_a \setminus \{a\}),$$

where $\phi(R, C)$ is the capacity of a minimum s-t-cut in $(V, A \setminus \cup_{a \in R} C_a)$.

The intuition behind this definition is the following: For every $a \in R$, all flow that crossed the cut C_a must cross the $C_a \setminus \{a\}$ if a fails. If a flow path does not cross any cut in C_a, then it crosses the minimum s-t-cut in $(V, A \setminus \cup_{a \in R} C_a)$. Therefore the capacity of an R-cut is an upper bound on the value of any reroutable flow.

▶ **Lemma 6.** $\mathrm{val}(x) \leq \mathrm{cap}(R, C)$ *for any reroutable flow* x *and any* R*-cut* (R, C).

It can be shown that R-cuts correspond to integral solutions to the dual of $[\mathrm{LP}_{\mathrm{strict}}]$. We now give a constructive proof bounding the duality gap between maximum strictly reroutable flow and minimum R-cut (or, equivalently, the integrality gap of the dual LP). In Section 2.4 we give an example showing that the bound is tight.

▶ **Theorem 7.** *Let* x *be a strictly reroutable flow of maximum value and let* (R, C) *be an* R*-cut of minimum capacity. Then* $\mathrm{val}(x) \geq \frac{1}{2}\,\mathrm{cap}(R, C)$.

Proof. For $a \in A$, let C_a be minimum $\mathrm{tail}(a)$-t-cut in D containing a and define $u'(a) := \min\{u(a), u(C_a \setminus \{a\})\}$. Let C' be a minimum s-t-cut in D with respect to the capacities u' and let x' be a corresponding maximum flow. Now define $R := \{a \in C' : u'(a) < u(a)\}$. Observe that R and $(C_a)_{a \in R}$ define an R-cut and that $\phi(R, C) \leq u(C' \setminus R)$. We obtain

$$\mathrm{cap}(R, C) \leq \sum_{a \in C' \setminus R} u(a) + \sum_{a \in R} u(C_a \setminus \{a\}) = \sum_{a \in C'} u'(a) = \mathrm{val}(x').$$

Now let $x := x'/2$. It is sufficient to show that x is a strictly reroutable flow. By contradiction assume that there is $\bar{a} \in A$ for which there is no strict rerouting of x. By the max flow/min cut theorem, there must be a $\mathrm{tail}(\bar{a})$-t-cut $\bar{C}$ in $(V, A \setminus \{\bar{a}\})$ with $\sum_{a \in \bar{C}} \bar{u}_x(a) < x(\bar{a})$. Note that $x(a) \leq u'(a)/2 \leq u(a)/2$ for every $a \in A$ by construction of x. Thus

$$\frac{1}{2} \sum_{a \in \bar{C}} u(a) \leq \sum_{a \in \bar{C}} (u(a) - x(a)) < x(\bar{a}) \leq \frac{1}{2} u'(\bar{a}) \leq \frac{1}{2} u(C_{\bar{a}} \setminus \{\bar{a}\}).$$

However, this implies that $\bar{C} \cup \{\bar{a}\}$ is a smaller $\mathrm{tail}(\bar{a})$-t-cut than $C_{\bar{a}}$, a contradiction. ◀

Computing a minimum capacity R-cut. Let us denote the problem of finding an R-cut of minimum capacity by Min R-Cut. The proof of Theorem 7 describes how to compute a 2-approximate solution to this problem.

▶ **Corollary 8.** *There is a* 2*-approximation algorithm for* Min R-Cut.

2.4 Summary of the bounds and tightness

Putting the bounds from Lemma 4 and Theorem 7 together, we obtain the following corollary.

▶ **Corollary 9.** *Let* (R, C) *be a minimum capacity* R*-cut and let* x_{RF} *and* x_{SRF} *be maximal reroutable and strictly reroutable flows, respectively. Then*

$$\mathrm{val}(x_{\mathrm{RF}}) \leq \mathrm{cap}(R, C) \leq 2\,\mathrm{val}(x_{\mathrm{SRF}}) \leq 2\,\mathrm{val}(x_{\mathrm{RF}}).$$

The example given in Figure 1 shows that each of the bounds proven in this section is tight. It also shows that optimal solutions to both Max RF and Max SRF can be fractional, even when capacities are integral. In the depicted network, and in further examples and reductions throughout the paper, we use the following gadget.

Backup links. A *backup link* from v to w is a v-w-path (a', a'') of length 2 in which the intermediate node is incident only to the two arcs of the path and $u(a') := u(a'') := \max_{a \in A} u(a)$. Note that $x(a') = x(a'') = 0$ for any reroutable flow, because when a'' fails, there is no $\mathrm{tail}(a'')$-t-path for rerouting the flow on that arc. A *bidirected* backup link between v and w consists of two distinct backup links, one from v to w and one from w to v.

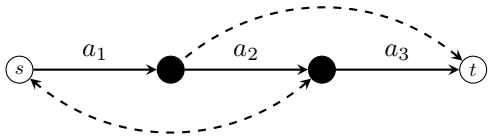


Figure 1 Example showing that the bounds given in Lemma 4 and Theorem 7 are tight. Dashed arcs correspond to (bidirected) backup links, which can only be used for rerouting. When all arcs have unit capacities, the maximum (strictly) reroutable flow has a value of 1/2. When changing the capacity of a_1 to 2, the maximum reroutable flow value increases to 1, whereas the maximum strictly reroutable flow value remains 1/2. The minimum R-cut capacity is 1 in both cases.

▶ Remark. Note that the worst-case for the bounds in Corollary 9 cannot be attained simultaneously, i.e., in any given instance either the max flow/min cut gap or the gap between reroutable and strictly reroutable flow has to be significantly smaller than 2—in fact, at least one of them has to be within $\sqrt{2}$.

3 Unit capacity networks

Throughout this section, we assume $u \equiv 1$. We will show that in this case, any reroutable flow can be transformed into a strictly reroutable flow of the same value. We start by giving an alternative characterization for strictly reroutable flows in unit capacity networks.

Cuts separating t. For $S \subseteq V$, let $\delta^+(S) := \{a \in A : \text{tail}(a) \in S, \text{head}(a) \in V \setminus S\}$ denote the cut induced by S. We define $\mathcal{S} := \{S \subset V \setminus \{t\} : S \neq \emptyset\}$ and let $\mathcal{C} := \{\delta^+(S) : S \in \mathcal{S}\}$ be the set of *t-separating cuts*. W.l.o.g. we assume $\delta^+(S) \neq \emptyset$ for all $S \in \mathcal{S}$, as no vertex in a set S with $\delta^+(S) = \emptyset$ can be on an s-t-path.

▶ **Lemma 10.** *Let x be an s-t-flow for capacities $u \equiv 1$. Then x is strictly reroutable if and only if $\sum_{a \in C}(1 - x(a)) \geq 1$ for all $C \in \mathcal{C}$.*

In the following, we identify those cuts that might violate the condition given in Lemma 10 for a (non-strictly) reroutable flow. We then show that this class of cuts forms a semi-lattice. This allows us to apply an uncrossing of the flow paths that iteratively eliminates the problematic cuts while maintaining reroutability.

Bad cuts. Let x be an s-t-flow and let $C \in \mathcal{C}$ be a t-separating cut. An arc $\bar{a} \in C$ is *(x, C)-bad* if there is an arc $a \in C$ and a path $P \in \mathcal{P}_{\bar{a} \to a}$ with $x(P) > 0$. A cut C is *x-bad* if all arcs $\bar{a} \in C$ are (x, C)-bad.

▶ **Lemma 11.** *Let x be a reroutable flow for capacities $u \equiv 1$. Let $C \in \mathcal{C}$ be a t-separating cut. If $\sum_{a \in C}(1 - x(a)) < 1$ then C is x-bad.*

Proof. By contradiction assume C is not x-bad. Then there must be an arc $\bar{a} \in C$ that is not (x, C)-bad. This implies that $\sum_{P \in \mathcal{P}_{\bar{a} \to a}} x(P) = 0$ for every $a \in C \setminus \{\bar{a}\}$. In particular, $\bar{u}_{x,\bar{a}}(a) = \bar{u}_x(a) = 1 - x(a)$ for all $a \in C \setminus \{\bar{a}\}$. Since all flow in the rerouting of x for failure of $\bar{a}$ needs to cross $C \setminus \{\bar{a}\}$, we obtain $\sum_{a \in C \setminus \{\bar{a}\}} \bar{u}_{x,\bar{a}}(a) \geq x(\bar{a})$. Adding $1 - x(\bar{a})$ to both sides of this inequality yields a contradiction. ◀

▶ **Lemma 12.** *Let x be a flow and let $S, S' \in \mathcal{S}$ be such that $\delta^+(S)$ and $\delta^+(S')$ are both x-bad. Then $\delta^+(S \cup S')$ is an x-bad t-separating cut as well.*

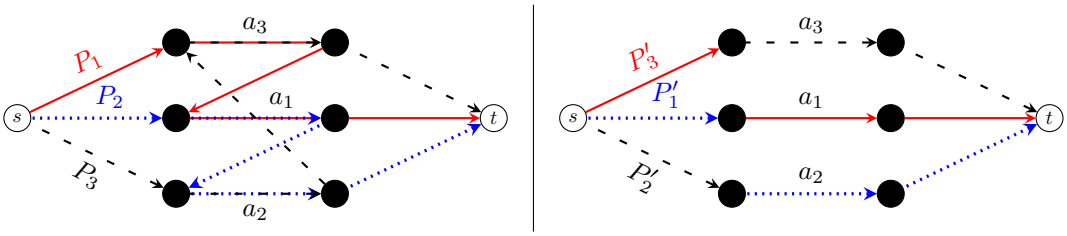


Figure 2 Uncrossing of paths on a bad cut.

Uncrossing paths. Let $P \in \mathcal{P}$. For two nodes $v, w \in V$ visited by P (in that order), we let $P[v, w]$ denote the subpath of path P starting at v and ending at w. Given another path $Q \in \mathcal{P}$ and an arc $a \in P \cap Q$, let $P \times_a Q$ be a simple s-t-path in the concatenation of $P[s, \text{head}(a)]$ and $Q[\text{head}(a), t]$.

▶ **Theorem 13.** *Let x be a reroutable flow for capacities $u \equiv 1$. Then there is a strictly reroutable flow x' with $\text{val}(x') = \text{val}(x)$ and $x'(a) \leq x(a)$ for all $a \in A$.*

Sketch of Proof. If x is not strictly reroutable, then by Lemmas 10 and 11 there must be an x-bad cut. By Lemma 12, there is a "rightmost" x-bad cut $C^* := \delta^+(S^*)$ where S^* is the union of all vertex sets defining x-bad cuts. Because C^* is bad, we obtain flow-carrying paths $P_1, \ldots, P_k$ and arcs $a_1, \ldots, a_k$ such that $a_i \in P_i \cap P_{i+1}$ is the last arc of P_i that crosses C^* for each $i \in [k]$ (with $P_{k+1} := P_1$). See Figure 2 for an illustration.

We uncross these paths by defining $P'_i := P_{i+1} \times_{a_i} P_i$ for $i \in [k]$. We obtain a new flow x' by decreasing the flow on all paths P_i by $\varepsilon := \min_i x(P_i)$ and increasing the flow on paths P'_i by ε for all $i \in [k]$. Observe that $\text{val}(x') = \text{val}(x)$ and $x'(a) \leq x(a)$ for all $a \in A$. We show that x' is also a reroutable flow. To this end, let $\bar{a} \in A$ and let $S \subseteq V \setminus \{t\}$ with $\text{tail}(\bar{a}) \in S$ and define $C := \delta^+(S)$. If $S \not\subseteq S^*$, then C is not x-bad by construction of S^*. In this case, it is easy to show that $\sum_{a \in C \setminus \{\bar{a}\}} \bar{u}_{x',\bar{a}}(a) \geq x'(\bar{a})$. If $S \subseteq S^*$, then a careful analysis shows that $\bar{u}_{x',\bar{a}}(a) \geq \bar{u}_{x,\bar{a}}(a)$ for all $a \in C$. Thus in both cases there is sufficient capacity to reroute flow when $\bar{a}$ fails. We repeat this procedure until we arrive at a strictly reroutable flow. ◀

▶ Remark. The proof of Theorem 13 preserves integrality. More specifically, if $x(P)$ is an integer multiple of α for every $P \in \mathcal{P}$, then x' can be chosen such that also $x'(P)$ is an integer multiple of α for every $P \in \mathcal{P}$.

▶ Remark. The characterization of strictly reroutable flows for unit capacities given in Lemma 10 can be extended to instances with arbitrary capacities. However, in the general case, a non-strictly reroutable flow might not have a bad cut.

4 Computing (half-)integral solutions

In some application contexts, flow cannot be split into arbitrarily small pieces. This is the setting we consider in this section. We say a flow x is *integral*, if $x(P) \in \mathbb{Z}$ for all $P \in \mathcal{P}$. We say that x is *half-integral* if $2x$ is integral.

For many fundamental flow problems, such as Max Flow or Min Cost Flow, integrality comes for free, i.e., as long as capacities are integral, there exists an optimal integral solution. In the case of reroutable flows, this property does not hold, see, e.g., Figure 3. In fact, it turns out to be *NP*-hard to decide whether there is a non-zero integral reroutable flow in a network.

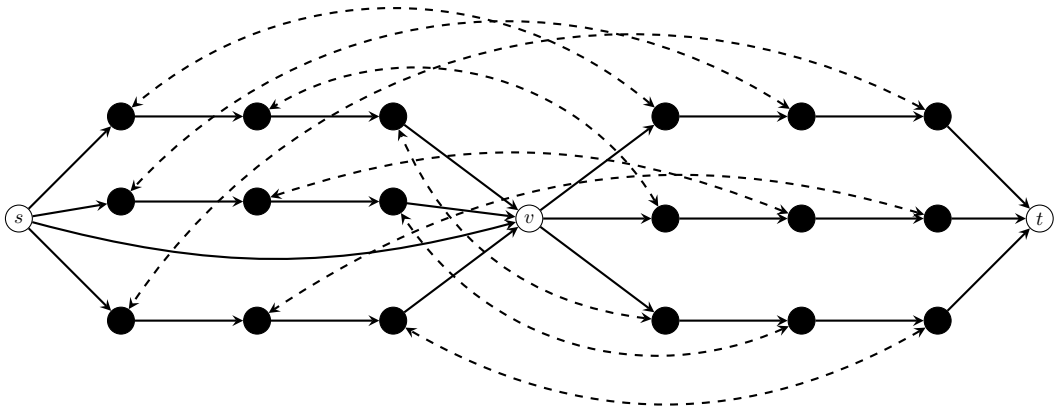


Figure 3 Example network in which no integral or half-integral reroutable flow is optimal. Dashed arcs represent bidirected backup links (see Section 2.4), all arcs have unit capacities. The maximum reroutable flow value is 2. This can only be achieved when $x(s,v) = 1$, the three s-v-paths all carry $1/3$ unit of flow, and the three v-t-paths all carry $2/3$ unit of flow.

▶ **Theorem 14.** *It is NP-hard to decide whether there is an integral (strictly) reroutable flow of value* 1*, even when restricted to instances with* $u \equiv 1$*.*

Note that this problem corresponds to sending a unit of flow along a single s-t-path. The hardness stems from a problem named Forbidden Pairs s-t-Path, which we introduce in Section 5. While it seems that Theorem 14 does not give much space for positive algorithmic results, we can do much better if we relax the integrality requirement slightly.

▶ **Theorem 15.** *Given a network with* $u \equiv 1$*, the algorithm given in Listing1 computes in polynomial time either a half-integral strictly reroutable flow of value* 1*, or correctly determines that no reroutable flow of value* 1 *exists.*

In particular, this implies that if we are interested in sending a single unit of flow, we never need to split our flow in more than two paths. Before we discuss the algorithm from Theorem 15, let us shortly discuss the case of arbitrary capacities. As a consequence of the max flow/min cut result proven in Section 2.3, we obtain the following approximation.

▶ **Theorem 16.** *If* u *is integral, then there is a strictly reroutable half-integral flow* x *with* $\mathrm{val}(x) \geq \mathrm{OPT}/2$*, where* OPT *is the value of a maximum reroutable flow. The flow* x *can be computed in polynomial time.*

Proof. Recall that in the proof of Theorem 7 we computed an s-t-flow x' that was maximal with respect to capacities $u'(a) := \min\{u(a), u(C_a \setminus \{a\})\}$. We then showed that the flow $x := x'/2$ is strictly reroutable and within a factor of 2 of a corresponding R-cut. In particular, $\mathrm{val}(x)$ is within a factor of 2 of the maximum reroutable flow value. Note that if u is integral, also u' is integral, and hence we can choose x' to be integral, ensuring that x is half-integral. ◀

Algorithm for computing a half-integral flow for unit demand

A natural starting point for an algorithm is to identify arcs $a \in A$ such that $\mathrm{tail}(a)$ is disconnected from t in $(V, A \setminus \{a\})$. Obviously, no reroutable flow can send a positive amount

Listing 1 Computing a half-integral reroutable unit demand flow

```
A_0 := ∅, A_1 := ∅
while ∃ a ∈ A \ A_0 : A_1 ∪ {a} is a tail(a)-t-cut in D
  A_0 := A_0 ∪ {a}
  A_1 ← {a' : a' is an s-t-bridge in (V, A \ A_0)}
end while
if A_0 is an s-t-cut in D
  return "No reroutable flow of value 1 exists."
else
  Let P_1, P_2 be two s-t-paths in (V, A \ A_0) such that P_1 ∩ P_2 = A_1.
  Let x be the flow defined by x(P_1) = x(P_2) = 1/2.
  return x
end if
```

of flow along such arcs, as after failure of a, the flow cannot be rerouted to t. Surprisingly, this simple preprocessing step can be generalized to an iterative procedure that solves the problem.

The algorithm, which is formally given in Listing 1, maintains two sets A_0 and A_1. In every iteration, it identifies an arc that cannot carry any flow in any reroutable flow and adds it to A_0. The set A_1 contains the s-t-bridges in the graph $(V, A \setminus A_0)$, i.e., all arcs whose removal disconnects s from t in that graph. Clearly, if $x(a) = 0$ for all $a \in A_0$, then every arc in A_1 must carry 1 unit of flow. If at some point A_0 becomes an s-t-cut, we know that no reroutable flow of value 1 exists. On the other hand, if the algorithm finds no more arcs to add to A_0 while s and t are still connected in $(V, A \setminus A_0)$, it computes two paths P_1, P_2 that only intersect at the bridges, and sends $1/2$ units of flow along each of them.

Proof of Theorem 15. To see that Algorithm 1 terminates in polynomial time, observe that $|A_0|$ is increased in every iteration of the while-loop and the loop thus terminates after at most $|A|$ iterations, each of which can be carried out in polynomial time.

Case 1: No flow exists. We now show that if Algorithm 1 denies the existence of a reroutable flow of value 1, this is indeed correct. By contradiction assume A_0 contains an s-t-cut but there exists a reroutable flow x of value 1. We prove by induction that at any step of algorithm the set A_0 fulfills the property that $x(a) = 0$ for all $a \in A_0$, yielding a contradiction. The claim is clearly true initially, when $A_0 = \emptyset$. Now consider any iteration of the while-loop, considering arc a. By induction hypothesis, every s-t-path P with $x(P) > 0$ must be a path in $(V, A \setminus A_0)$. Note that there is an order $a_1, \ldots, a_\ell$ of the set A_1 of s-t-bridges of $(V, A \setminus A_0)$ such that every such flow-carrying path contains all of these bridges in exactly that order. In particular $x(a_1) = \ldots = x(a_\ell) = 1$. Now consider the next arc a added to A_0 and assume by contradiction that $x(a) > 0$. By choice of a there is a tail(a)-t-cut $C \subseteq A_1 \cup \{a\}$ in D. Note that if $C \cap A_1 = \emptyset$, there is no rerouting of x in case of failure of arc a, as there is no tail(a)-t-path in $(V, A \setminus \{a\})$. Thus, let $a_k \in C \cap A_1$ be the bridge with the highest index k on the cut. We distinguish two cases:

- **(i)** Assume a appears before a_k on every flow-carrying path. Note that C is a tail(a_k)-t-cut because $a_k \in C$ and that $\sum_{a' \in C} \bar{u}_{x,a_k}(a') = 1 - x(a) < 1$. Therefore, the one unit of flow on a_k cannot be rerouted when a_k fails.
- **(ii)** Now assume a occurs after a_k on every flow-carrying path. But then, when a fails, the flow on a cannot be rerouted as all edges in $C \setminus \{a\} \subseteq A_1$ occur before a on every flow-carrying path and thus $\sum_{a' \in C \setminus \{a\}} \bar{u}_{x,a}(a') = 0$.

We thus deduce that $x(a) = 0$, completing the induction.

Case 2: Algorithm returns flow. Finally, we show that if $(V, A \setminus A_0)$ contains an s-t-path after completing the while-loop, then the flow x returned by the algorithm is a strictly reroutable flow. First observe that two s-t-paths P_1, P_2 in $(V, A \setminus A_0)$ with $P_1 \cap P_2 = A_1$ exist by the max flow/min cut theorem, as A_1 contains exactly the bridges of $(V, A \setminus A_0)$. Now consider the failure of any arc $\bar{a} \in A \setminus A_0$. Let C be a tail$(\bar{a})$-t-cut in D minimizing $U(C) := \sum_{a \in C \setminus \{\bar{a}\}} \bar{u}_x(a)$. We show that $U(C) \geq x(\bar{a})$, which by max flow/min cut implies that there is a rerouting of x in case of failure of $\bar{a}$. By termination condition of the while-loop, there is at least one arc $a' \in C \setminus (A_1 \cup \{\bar{a}\})$. Note that $x(a') \in \{0, 1/2\}$ and thus $U(C) \geq 1/2$. If $\bar{a} \notin A_1$, then $x(\bar{a}) \leq 1/2 \leq U(C)$. If $\bar{a} \in A_1$, we distinguish two cases.

(i) If $x(a') = 0$ then $U(C) \geq 1$ and the one unit of flow on $\bar{a}$ can be rerouted.

(ii) If $x(a') = 1/2$, then $a' \notin A_0$. Note that C is a tail(a')-t-cut in D and thus there is $a'' \in C \setminus A_1 \cup \{a'\}$ by termination condition of the while-loop. Note that, because $a'' \notin A_1$, we have $a'' \neq a$ and $x(a'') \leq 1/2$. Thus $U(C) \geq 1$ also in this last case.

We conclude that x is indeed strictly reroutable. ◀

▶ Remark. Note that our proof of Theorem 15 does not make use of Theorem 13. Instead, it gives a simple alternative argument for the equivalence of reroutable and strictly reroutable flows in unit capacity networks, for the special case of unit value flows.

▶ Remark. Theorem 15 implies that, for networks with $u \equiv 1$, if there exists any reroutable flow of value 1, then there exists a half-integral strictly reroutable flow of value 1. The example given in Figure 3, however, reveals that this is no longer true for flows of higher value, as the unique maximum reroutable flow uses paths with flow value 1/3.

5 Hardness results

In this section, we give hardness results for Max RF and some variants of the problem.

Paths avoiding forbidden pairs. Our hardness results are based on reductions from Forbidden Pairs s-t-Path, which is defined as follows: We are given a digraph $D' = (V', A')$, two nodes $s', t' \in V'$, and a set of forbidden arc pairs $\mathcal{F} \subseteq \{\{a, \bar{a}\} : a, \bar{a} \in A\}$. The task is to find an s'-t'-path P that does not contain both arcs of any pair, i.e., $|S \cap P| \leq 1$ for all $S \in \mathcal{F}$. It is not hard to see that Forbidden Pairs s-t-Path is *NP*-hard [12].

5.1 General capacities

▶ **Theorem 3.** Max RF *is NP-hard, even when* $u(a) \in \{1, 2\}$ *for all* $a \in A$.

Sketch of Proof. We construct a gadget that allows us to introduce forbidden pairs for flow paths in the network. Starting with an instance of Forbidden Pairs s-t-Path, we append a sequence of parallel length-2 paths (g_i, h_i) and $(\bar{g}_i, \bar{h}_i)$, one pair of paths for each pair $\{a_i, \bar{a}_i\} \in \mathcal{F}$, leading from t' to the new sink t. For each i, we connect a_i and h_i, and $\bar{a}_i$ and $\bar{h}_i$, respectively, with bidirected backup links. The construction is depicted in Figure 4. In a reroutable s-t-flow of value 2, both h_i and $\bar{h}_i$ are saturated. When a_i fails, the only path for rerouting leads via h_i, hence all flow that traverses a_i must be on paths in $\mathcal{P}_{a_i \to h_i}$. Likewise all flow that traverses $\bar{a}_i$ must be on paths in $\mathcal{P}_{\bar{a}_i \to \bar{h}_i}$. As $\mathcal{P}_{h_i} \cap \mathcal{P}_{\bar{h}_i} = \emptyset$, no flow-carrying path can use both a_i and $\bar{a}_i$ for any i. Thus if a reroutable flow of value 2 exists, there is a path avoiding the forbidden pairs. For the converse of this argument, it is important that $u(a_i) = u(\bar{a}_i) = 2$. This allows for a rerouting when h_i or $\bar{h}_i$ fails. ◀

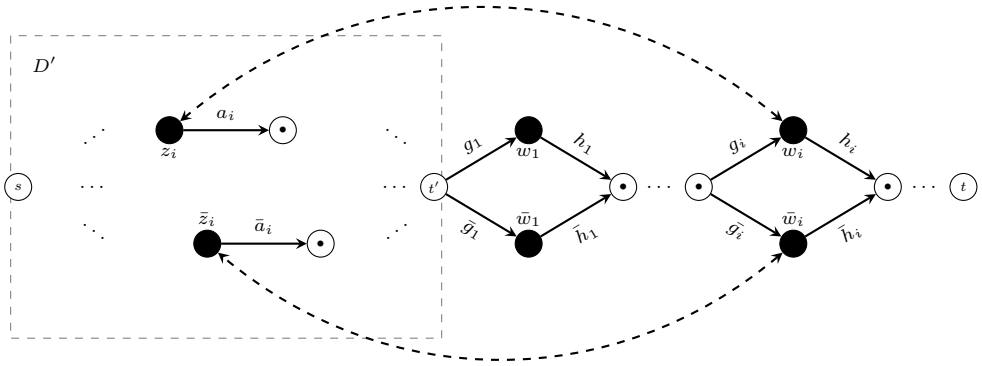


Figure 4 Construction for the proof of Theorem 3. The dashed box contains the graph D' from the Forbidden Pairs s-t-Path instance. The arcs $a_i, \bar{a}_i$ have capacity 2 for all i, all other arcs have unit capacity. In a reroutable flow of value 2, the arcs h_i and $\bar{h}_i$ must be saturated for all i. Any rerouting for a_i has to traverse h_i and any rerouting for $\bar{a}_i$ has to traverse $\bar{h}_i$.

5.2 Multiple arc failures

A natural generalization of Max RF and Max SRF allows multiple simultaneous arc failures. When a set of arcs S fails, flow is interrupted where it first encounters an arc from S and has to be rerouted from that point to the sink. A flow is (strictly) k-reroutable, if there is a rerouting for any failure of a set $S \subseteq A$ with $|S| \leq k$. We denote the corresponding problem of finding a (strictly) k-reroutable flow of maximum value by Max (Strictly) k-Reroutable Flow. It turns out that dealing even with only 2 arc failures in unit capacity networks is *NP*-hard in both cases.

▶ **Theorem 17.** Max (Strictly) k-Reroutable Flow *is NP-hard, even when restricted to instances with* $k = 2$ *and* $u \equiv 1$.

Acknowledgments. We thank David Adjiashvili and Marco Senatore for helpful discussions.

References

1 David Adjiashvili, Gianpaolo Oriolo, and Marco Senatore. The online replacement path problem. In *Algorithms – ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.
2 Charu C. Aggarwal and James B. Orlin. On multiroute maximum flows in networks. *Networks*, 39(1):43–52, 2002.
3 Y. P. Aneja, R. Chandrasekaran, and K. P. K. Nair. Maximizing residual flow under an arc destruction. *Networks*, 38(4):194–198, 2001.
4 Dimitris Bertsimas, Ebrahim Nasrabadi, and James B. Orlin. On the power of randomization in network interdiction. *Operations Research Letters*, 44(1):114–120, 2016.
5 Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and adaptive network flows. *Operations Research*, 61:1218–1242, 2013.
6 Graham Brightwell, Gianpaolo Oriolo, and F. Bruce Shepherd. Reserving resilient capacity in a network. *SIAM Journal on Discrete Mathematics*, 14(4):524–539, 2001.
7 Chandra Chekuri, Anupam Gupta, Amit Kumar, Joseph Naor, and Danny Raz. Building edge-failure resilient networks. *Algorithmica*, 43(1-2):17–41, 2005.

8 Stephen R. Chestnut and Rico Zenklusen. Hardness and approximation for network flow interdiction. *Networks*, 2017.

9 Amaro de Sousa and Gil Soares. Improving load balance and minimizing service disruption on ethernet networks with IEEE 802.1 S MSTP. In *Workshop on IP QoS and Traffic Control*, pages 25–35, 2007.

10 Yann Disser and Jannik Matuschke. The complexity of computing a robust flow, 2017.

11 Lester R. Ford and Delbert R. Fulkerson. *Flows in networks*. Princeton Univ. Press, 1962.

12 Harold N. Gabow, Shachindra N. Maheshwari, and Leon J. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, SE-2(3):227–231, 1976.

13 Fabrizio Grandoni, Gaia Nicosia, Gianpaolo Oriolo, and Laura Sanità. Stable routing under the spanning tree protocol. *Operations Research Letters*, 38(5):399–404, 2010.

14 Alan J. Hoffman. A generalization of max flow—min cut. *Mathematical Programming*, 6(1):352–359, 1974.

15 Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999.

16 Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.

17 Steven J. Phillips and Jeffery R. Westbrook. Approximation algorithms for restoration capacity planning. In *Algorithms – ESA'99*, volume 1643 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 1999.

18 F. Bruce Shepherd. Single-sink multicommodity flow with side constraints. In *Research Trends in Combinatorial Optimization*, pages 429–450. Springer, 2009.

The Parameterized Complexity of Positional Games*

Édouard Bonnet[1], Serge Gaspers[2], Antonin Lambilliotte[3], Stefan Rümmele[4], and Abdallah Saffidine[5]

1 **Middlesex University, London, UK**
edouard.bonnet@dauphine.fr
2 **The University of New South Wales, Sydney, Australia; and Data61, CSIRO, Sydney, Australia**
sergeg@cse.unsw.edu.au
3 **École Normale Supérieure de Lyon, Lyon, France**
antonin.lambilliotte@ens-lyon.fr
4 **The University of New South Wales, Sydney, Australia; and The University of Sydney, Sydney, Australia**
s.rummele@unsw.edu.au
5 **The University of New South Wales, Sydney, Australia**
abdallah.saffidine@gmail.com

Abstract

We study the parameterized complexity of several positional games. Our main result is that Short Generalized Hex is W[1]-complete parameterized by the number of moves. This solves an open problem from Downey and Fellows' influential list of open problems from 1999. Previously, the problem was thought of as a natural candidate for AW[*]-completeness. Our main tool is a new fragment of first-order logic where universally quantified variables only occur in inequalities. We show that model-checking on arbitrary relational structures for a formula in this fragment is W[1]-complete when parameterized by formula size.

We also consider a general framework where a positional game is represented as a hypergraph and two players alternately pick vertices. In a Maker-Maker game, the first player to have picked all the vertices of some hyperedge wins the game. In a Maker-Breaker game, the first player wins if she picks all the vertices of some hyperedge, and the second player wins otherwise. In an Enforcer-Avoider game, the first player wins if the second player picks all the vertices of some hyperedge, and the second player wins otherwise.

Short Maker-Maker, Short Maker-Breaker, and Short Enforcer-Avoider are respectively AW[*]-, W[1]-, and co-W[1]-complete parameterized by the number of moves. This suggests a rough parameterized complexity categorization into positional games that are complete for the first level of the W-hierarchy when the winning condition only depends on which vertices one player has been able to pick, but AW[*]-complete when it depends on which vertices both players have picked. However, some positional games with highly structured board and winning configurations are fixed-parameter tractable. We give another example of such a game, Short k-Connect, which is fixed-parameter tractable when parameterized by the number of moves.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Hex, Maker-Maker games, Maker-Breaker games, Enforcer-Avoider games, parameterized complexity theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.90

* Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048). Abdallah Saffidine is the recipient of an ARC DECRA Fellowship (DE150101351). This work received support under the ARC's Discovery Projects funding scheme (DP150101134).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 90; pp. 90:1–90:14

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

1 Introduction

In a *positional game* [13], two players alternately claim unoccupied elements of the board of the game. The goal of a player is to claim a set of elements that form a winning set, and/or to prevent the other player from doing so.

Tic-Tac-Toe, its competitive variant played on a 15×15 board, Gomoku, as well as Hex are the most well-known positional games. When the size of the board is not fixed, the decision problem, whether the first player has a winning strategy from a given position in the game is PSPACE-complete for many such games. The first result was established for Generalized Hex, a variant played on an arbitrary graph [8]. Reisch [15] soon followed up with results for gomoku [15] and Hex played on a board [16]. More recently, PSPACE-completeness was obtained for Havannah [4] and several variants of Connect(m, n, k, p, q) [14], a framework that encompasses Tic-Tac-Toe and Gomoku.

In a Maker-Maker game, also known as strong positional game, the winner is the first player to claim all the elements of some winning set. In a Maker-Breaker game, also known as weak positional game, the first player, Maker, wins by claiming all the elements of a winning set, and the second player, Breaker, wins by preventing Maker from doing so. In an Enforcer-Avoider game, the first player, Enforcer, wins if the second player claims all the vertices of a winning set, and the second player, Avoider, wins otherwise.

In this paper, we consider the corresponding short games, of deciding whether the first player has a winning strategy in ℓ moves from a given position in the game, and parameterize them by ℓ. The parameterized complexity of short games is known for games such as generalized chess [19], generalized geography [1, 2], and pursuit-evasion games [20]. For Hex, played on a hexagonal grid, the short game is FPT and for Generalized Hex, played on an arbitrary graph, the short game is W[1]-hard and in AW[*].

When winning sets are given as arbitrary hyperedges in a hypergraph, we refer to the three game variants as Maker-Maker, Maker-Breaker, and Enforcer-Avoider, respectively. Maker-Breaker was first shown PSPACE-complete by Schaefer [17] under the name G_{pos}(POS DNF). A simpler proof was later given by Byskov [5] who also showed PSPACE-completeness of Maker-Maker. To the best of our knowledge, the classical complexity of Enforcer-Avoider has not been established yet.

We will show that the short game for Generalized Hex is W[1]-complete, solving an open problem stated numerous times [4, 7, 6, 10, 18], we establish that the short game for a generalization of Tic-Tac-Toe is FPT, and we determine the parameterized complexity of the short games for Maker-Maker, Maker-Breaker, and Enforcer-Avoider. One of our main tools is a new fragment of first-order logic where universally-quantified variables only occur in inequalities and no other relations. After giving some necessary definitions in the next section, we will state our results precisely, and discuss them. The rest of the paper is devoted to the proofs, with some parts deferred to the long version [3].

2 Preliminaries

Finite structures. A vocabulary τ is a finite set of relation symbols, each having an associated arity. A finite structure $\mathcal{A}$ over τ consists of a finite set A, called the universe, and for each R in τ a relation over A of corresponding arity. An (undirected) graph is a finite structure $G = (V, E)$, where E is a symmetric binary relation. A hypergraph is a finite structure $G = (V \cup E, IN)$, where $IN \subseteq V \times E$ is the incidence relation between vertices and edges. Sometimes it is more convenient to denote a hypergraph instead by a tuple $G = (V, E)$ where E is a set of subsets of V.

First-order logic. We assume a countably infinite set of variables. Atomic formulas over vocabulary τ are of the form $x_1 = x_2$ or $R(x_1, \ldots, x_k)$ where $R \in \tau$ and $x_1, \ldots, x_k$ are variables. The class FO of all first-order formulas over τ consists of formulas that are constructed from atomic formulas over τ using standard Boolean connectives $\neg, \wedge, \vee$ as well as quantifiers $\exists, \forall$ followed by a variable. Let φ be a first-order formula. The size of (a reasonable encoding of) φ is denoted by $|\varphi|$. The variables of φ that are not in the scope of a quantifier are called free variables. We denote by $\varphi(\mathcal{A})$ the set of all assignments of elements of A to the free variables of φ such that φ is satisfied. We call $\mathcal{A}$ a model of φ if $\varphi(\mathcal{A})$ is not empty. The class Σ_1 contains all first-order formulas of the form $\exists x_1, \ldots, \exists x_k \varphi$ where φ is a quantifier free first-order formula.

Parameterized complexity. The class FPT contains all parameterized problems that can be decided by an FPT-algorithm. An FPT-algorithm is an algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$, where $f(\cdot)$ is an arbitrary computable function that only depends on the parameter k and n is the size of the problem instance. An FPT-*reduction* of a parameterized problem Π to a parameterized problem Π' is an FPT-algorithm that transforms an instance (I, k) of Π to an instance (I', k') of Π' such that: (i) (I, k) is a yes-instance of Π if and only if (I', k') is a yes-instance of Π', and (ii) $k' = g(k)$, where $g(\cdot)$ is an arbitrary computable function that only depends on k. Hardness and completeness with respect to parameterized complexity classes is defined analogously to the concepts from classical complexity theory, using FPT-reductions. The following parameterized classes will be needed in this paper: FPT $\subseteq$ W[1] $\subseteq$ AW[*]. Many parameterized complexity classes can be defined via a version of the following model checking problem.

MC(Φ)
Instance: Finite structure $\mathcal{A}$ and formula $\varphi \in \Phi$.
Parameter: $\lvert\varphi\rvert$.
Problem: Decide whether $\varphi(\mathcal{A}) \neq \emptyset$.

In particular, the problem MC(Σ_1) is W[1]-complete and the problem MC(FO) is AW[*]-complete (see for example [9]).

Positional games. Positional games are played by two players on a hypergraph $G = (V, E)$. The vertex set V indicates the set of available positions, while the each hyperedge $e \in E$ denotes a winning configuration. For some games, the hyperedges are implicitly defined, instead of being explicitly part of the input. The two players alternatively claim unclaimed vertices of V until either all elements are claimed or one player wins. A *position* in a positional game is an allocation of vertices to the players, who have already claimed these vertices. The *empty position* is the position where no vertex is allocated to a player. The notion of winning depends on the game type. In a *Maker-Maker game*, the first player to claim all vertices of some hyperedge $e \in E$ wins. In a *Maker-Breaker game*, the first player (*Maker*) wins if she claims all vertices of some hyperedge $e \in E$. If the game ends and player 1 has not won, then the second player (*Breaker*) wins. In an *Enforcer-Avoider game*, the first player (*Enforcer*) wins if the second player (*Avoider*) claims all vertices of some hyperedge $e \in E$. If the game ends and player 1 has not won, then the second player wins. A positional game is called an ℓ-move game, if the game ends either after a player wins or both players played ℓ moves. A winning strategy for player 1 is a move for player 1 such that for all moves of player 2 there exists a move of player 1... such that player 1 wins.

3 Results

The first game we consider is a Maker-Maker game that generalizes well-known games Tic-Tac-Toe, Connect6, and Gomoku (also known as Five in a Row). In Connect(m, n, k, p, q), the vertices are cells of an $m \times n$ grid, each set of k aligned cells (horizontally, vertically, or diagonally) is a winning set, the first move by player 1 is to claim q vertices, and then the players alternate claim p unclaimed vertices at each turn. Tic-Tac-Toe corresponds to Connect$(3, 3, 3, 1, 1)$, Connect6 to Connect$(19, 19, 6, 2, 1)$, and Gomoku to Connect$(19, 19, 5, 1, 1)$. Variations with different board sizes are also common. In the Short k-Connect problem, the input is the set of $m \cdot n$ vertices, an assignment of some of these vertices to the two players, the integer p, and the parameter ℓ. The winning sets corresponding to the k aligned cells are implicitly defined. The question is whether player 1 has a winning strategy from this position in at most ℓ moves. We omit q from the problem definition of Short k-Connect since we are modeling games that advanced already past the initial moves. Our first result (proved in Section 4) is that Short k-Connect is fixed-parameter tractable for parameter ℓ. (In all our results, the parameter is the number of moves, ℓ.)

▶ **Theorem 1.** Short k-Connect *is* FPT.

The main reason for this tractability is the rather special structure of the winning sets. It helps reducing the problem to model checking for first-order logic on locally bounded treewidth structures, which is FPT [11].

A similar strategy was recently used to show that Short Hex is FPT [4]. The Hex game is played on a parallelogram board paved by hexagons, each player owns two opposite sides of the parallelogram. Players alternately claim an unclaimed cell, and the first player to connect their sides with a path of connected hexagons wins the game. Note that we may view Hex as a Maker-Breaker game: if the second player manages to disconnect the first players sides, he has created a path connecting his sides. Bonnet et al. [4] also considered a well-known generalization to arbitrary graphs. The Generalized Hex game is played on a graph with two specified vertices s and t. The two players alternately claim an unclaimed vertex of the graph, and player 1 wins if she can connect s and t by vertices claimed by her, and player 2 wins if he can prevent player 1 from doing so. The Short Generalized Hex problem has as input a graph G, two vertices s and t in G, an allocation of some of the vertices to the players, and an integer ℓ. The parameter is ℓ, and the question is whether player 1 has a winning strategy to connect s and t in ℓ moves.

The Short Generalized Hex problem is known to be in AW[*] and was conjectured to be AW[*]-complete [4, 7, 6, 10, 18]. In fact, AW[*] is thought of as the natural home for most short games [7], playing a similar role in parameterized complexity as PSPACE in classical complexity for games with polynomial length. However, Bonnet et al. [4] only managed to show that Short Generalized Hex is W[1]-hard, leaving a complexity gap between W[1] and AW[*]. Our next result is to show that Short Generalized Hex is in W[1]. Thus, Short Generalized Hex is in fact W[1]-complete.

▶ **Theorem 2.** Short Generalized Hex *is* W[1]*-complete.*

Our main tool is a new fragment of first-order logic for which model-checking on arbitrary relational structures is W[1]-complete parameterized by the length of the formula. This fragment, which we call $\forall^{\neq}$-FO, is the fragment of first-order logic where universally-quantified variables appear only in inequalities.

▶ **Theorem 3.** MC($\forall^{\neq}$-FO) *is* W[1]*-complete.*

This result is proved by reducing a formula in $\forall^{\neq}$-FO to a formula in Σ_1. The $\forall^{\neq}$-FO logic makes it convenient to express short games where we can express that player 1 can reach a certain configuration without being blocked by player 2, no matter what configurations player 2 reaches. This is indeed the case for Generalized Hex, where we are merely interested in knowing if player 1 can connect s and t without being blocked by player 2.

More generally, this is the case for Short Maker-Breaker, where the input is a hypergraph $G = (V, E)$, a position, and an integer ℓ, and the question is whether player 1 has a winning strategy to claim all the vertices of some hyperedge in ℓ moves.

▶ **Theorem 4.** Short Maker-Breaker *is* W[1]*-complete.*

The fact that Short Maker-Breaker is PSPACE-complete and W[1]-complete (and *not* AW[*]-complete) may challenge the intuition one has on alternation. Looking at the classical complexity (PSPACE-completeness), it seems that both players have comparable expressivity and impact over the game. As the game length is polynomially bounded, if the outcome could be determined by only guessing a sequence of moves from one player, then the problem would lie in NP. Now from the parameterized complexity standpoint, Short Maker-Breaker is equivalent under FPT reductions to guessing the k vertices of a clique (as in the seminal W[1]-complete k-Clique problem); no alternation there. Those considerations may explain why it was difficult to believe that Generalized Hex is *not* AW[*]-complete as conjectured repeatedly [18, 6, 7].

This is also in contrast to Short Maker-Maker, where the input is a hypergraph $G = (V, E)$, a position, and an integer ℓ, and the question is whether player 1 has a strategy to be the first player claiming all the vertices of some hyperedge in ℓ moves.

▶ **Theorem 5.** Short Maker-Maker *is* AW[*]*-complete.*

For the remaining type of positional games, the Short Enforcer-Avoider problem has as input a hypergraph $G = (V, E)$, a position, and an integer ℓ, and the question is whether player 1 has a strategy to claim ℓ vertices that forces player 2 to complete a hyperedge. Again, player 1 can only block some moves of player 2, and the winning condition for player 1 can be expressed in $\forall^{\neq}$-FO.

▶ **Theorem 6.** Short Enforcer-Avoider *is* co-W[1]*-complete.*

Our results suggest that a structured board may suggest that a positional game is FPT, but otherwise, the complexity depends on how the winning condition for player 1 can be expressed. If it only depends on what positions player 1 has reached, our results suggest that the problem is W[1]-complete, but when the winning condition for player 1 also depends on the position player 2 has reached, the game is probably AW[*]-complete.

4 Short k-Connect is FPT

Graph G represents an $m \times n$ board in the following sense. Every board cell is represented by a vertex. Horizontal, vertical and diagonal neighbouring cells are connected via an edge. Vertex sets V_1 and V_2 represent the vertices already occupied by Player 1 and Player 2. While integer p, the number of stones to be placed during a move, is part of the input, we restrict it to values below constant k as games with $p \geq k$ are trivial.

Short k-Connect
Instance: A graph $G = (V, E)$ representing an $m \times n$ board, occupied vertices $V_1, V_2 \subseteq V$, and integer p and ℓ.
Parameter: ℓ.
Problem: Decide whether Player 1 has a winning strategy with at most ℓ moves.

▶ **Theorem 1.** Short k-Connect *is* FPT.

Proof. We reduce Short k-Connect to first-order model checking MC(FO) on a bounded degree graph. Using a result by Seese [21], it follows that Short k-Connect is FPT. Let (G, V_1, V_2, p, ℓ) be an instance of Short k-Connect, where $G = (V, E)$. We construct instance $(\mathcal{A}, \varphi)$ of MC(FO) as follows. Let *EDGE* be a binary relation symbol and let *V1* and *V2* be unary relation symbols. Then $\mathcal{A}$ is the $\{EDGE, V1, V2\}$-structure $(V, EDGE^{\mathcal{A}}, V1^{\mathcal{A}}, V2^{\mathcal{A}})$ with $EDGE^{\mathcal{A}} \coloneqq E$, $V1^{\mathcal{A}} \coloneqq V_1$, and $V2^{\mathcal{A}} \coloneqq V_2$. FO-formula φ is defined as $\varphi \equiv \exists x_1^1 \exists x_1^2 \dots \exists x_1^p \forall y_1^1 \dots \forall y_1^p \exists x_2^1 \dots \exists x_2^p \forall y_2^1 \dots \exists x_\ell^p \exists u_1 \exists u_2 \dots \exists u_k \forall v_1 \forall v_2 \dots \forall v_k \psi$,

$$\psi \equiv \bigvee_{i=0}^{\ell} \Big[\mathit{legalP1}_i(x_1^1, \dots, x_1^p, y_1^1, \dots, x_\ell^p) \wedge \Big(\neg \mathit{legalP2}_i(x_1^1, \dots, x_1^p, y_1^1, \dots, x_\ell^p) \vee \Big(\mathit{configP1}_i(x_1^1, \dots, x_\ell^p, u_1, \dots, u_k) \wedge \bigwedge_{j=1}^{k-2} \mathit{aligned}(u_j, u_{j+1}, u_{j+2}) \wedge \Big(\neg \mathit{configP2}_i(y_1^1, \dots, y_\ell^p, v_1, \dots, v_k) \vee \neg \bigwedge_{j=1}^{k-2} \mathit{aligned}(v_j, v_{j+1}, v_{j+2}) \Big) \Big) \Big) \Big],$$

$$\begin{aligned}
\mathit{path}(u, v, w) &\equiv \mathit{EDGE}(u, v) \wedge \mathit{EDGE}(v, w), \\
\mathit{hor_vert}(u, v, w) &\equiv \exists x \exists y\, \mathit{path}(u, v, w) \wedge \mathit{path}(u, x, w) \wedge \mathit{path}(u, y, w) \wedge \mathit{path}(x, v, y) \wedge \\
&\qquad \forall z \Big[\big(z \neq v \wedge z \neq x \wedge z \neq y \big) \rightarrow \neg \mathit{path}(u, z, w) \Big], \\
\mathit{diag}(u, v, w) &\equiv \mathit{path}(u, v, w) \wedge \forall x \Big[x \neq v \implies \neg \mathit{path}(u, x, w) \Big], \\
\mathit{aligned}(u, v, w) &\equiv \mathit{hor_vert}(u, v, w) \vee \mathit{diag}(u, v, w).
\end{aligned}$$

Variables x_i^j represent the jth stone in Player 1's ith move and variables y_i^j represent the jth stone in Player 2's ith move. The sequences $u_1 \dots u_k$ and $v_1 \dots v_k$ represent possible winning configurations for Player 1 and Player 2. The structure of ψ is the following. The first disjunction ranging from $i = 0$ to $i = \ell$ represents the number of moves Player 1 needs to win the game. We then ensure that the x variables represent legal moves by Player 1. Further, either variables y do not represent legal moves by Player 2, or Player 1 achieved a winning configuration. For the latter, we assure that variables u represent aligned vertices occupied by Player 1. Finally, we check that Player 2 did not achieve a winning configuration before, that is vertices v do not represent aligned vertices occupied by Player 2. Formula $\mathit{path}(u, v, w)$ expresses that there is a path of length 2 between vertices u and w via v ($\mathit{configP1}_i$ and $\mathit{configP2}_i$ ensure that the arguments are disjoint vertices). Formula $\mathit{hor\_vert}(u, v, w)$ expresses that vertices u, v, and w are aligned horizontally or vertically in this order. A case analysis shows that u, v and w are horizontally or vertically aligned if and only if there are exactly three nodes at distance 1 of u and w, and that v is in the middle of the other two. In case u, v and w are located on one of the border lines of the board, there are exactly two nodes at distance 1. Formula $\mathit{diag}(u, v, w)$ expresses that vertices u, v, and w are diagonally aligned in this order. This is the case if there exists no other length

2 path between u and w. Formula $\mathit{aligned}(u, v, w)$ expresses that vertices u, v, and w are aligned (in that order). Formula $\mathit{legalP1}_i$ (see [3]) ensures that variables x_i^j represent legal moves of Player 1, that is vertices not contained in V_1 or V_2 or previously played vertices. Analogously, $\mathit{legalP2}_i$ ensures that variables y_i^j represent legal moves of Player 2. Formula, $\mathit{configP1}_i$ (see [3]) expresses that variables $u_1, \ldots, u_k$ form a valid configuration of exactly k vertices out of the set of V_1 or vertices played by Player 1. Analogously, $\mathit{configP2}_i$ states that variables $v_1, \ldots, v_k$ form a valid configuration of exactly k vertices out of the set of V_2 or vertices played by Player 2. The size of φ is polynomial in ℓ, k, and p. Since k is a constant and p is bounded by k, we have an FO formula polynomial in our parameter ℓ. Graph G represents a grid with diagonals. Hence, G has maximum degree 8. It follows from Seese [21] that Short Connect is FPT. ◀

5 MC($\forall^{\neq}$-FO) is W[1]-complete

The class $\forall^{\neq}$-FO contains all first-order formulas of the form $Q_1x_1Q_2x_2Q_3x_3 \ldots Q_kx_k\varphi$, with $Q_i \in \{\forall, \exists\}$ and φ being a quantifier free first-order formula such that every $\forall$-quantified variable x_i only occurs in inequalities, that is in relations of the form $x_i \neq x_j$ for some variable x_j. Furthermore, φ does not contain any other variables besides $x_1, \ldots, x_k$.

▶ **Theorem 3.** MC($\forall^{\neq}$-FO) *is* W[1]*-complete.*

Proof. Hardness: Every Σ_1 formula is contained in the class $\forall^{\neq}$-FO. Hence, W[1]-hardness follows from W[1]-completeness of MC(Σ_1).

Membership: By reduction to MC(Σ_1). Let $(\mathcal{A}, \varphi)$ be an instance of MC($\forall^{\neq}$-FO). If φ contains only existential quantifiers then $(\mathcal{A}, \varphi)$ is already an instance of MC(Σ_1). Hence, let $\varphi = Q_1x_1Q_2x_2 \ldots Q_{i-1}x_{i-1}\forall x_i\exists x_{i+1}\exists x_{i+2} \ldots \exists x_k\psi$ with $Q_j \in \{\forall, \exists\}$ for $1 \leq j < i$, ψ is in negation normal form and $|\varphi| = \ell$. That is, x_i is the rightmost of the universal quantified variables. In order to reduce $(\mathcal{A}, \varphi)$ to an instance of MC(Σ_1), we need a way to remove all universal quantifications. We will show how to eliminate the universal quantification of x_i. This technique can then be used to iteratively eliminate all the universal quantifiers. Let $\varphi_1(x_1, \ldots, x_{i-1})$ be the subformula $\varphi_1(x_1, \ldots, x_{i-1}) = \forall x_i\exists x_{i+1} \ldots \exists x_k\psi$. We will show that we can replace $\varphi_1(x_1, \ldots, x_{i-1})$ by

$$\varphi_2(x_1, \ldots, x_{i-1}) = \exists y_i\exists y_{i+1} \ldots \exists y_k\Big(\psi[y_i/x_i, y_{i+1}/x_{i+1}, \ldots, y_k/x_k] \wedge \tag{1}$$

$$\bigwedge_{j=1}^{i-1} \exists y_{i+1}^j\exists y_{i+2}^j \ldots \exists y_k^j\psi[x_j/x_i, y_{i+1}^j/x_{i+1}, y_{i+2}^j/x_{i+2}, \ldots, y_k^j/x_k] \wedge \tag{2}$$

$$\bigwedge_{j=i+1}^{k} \exists y_{i+1}^j\exists y_{i+2}^j \ldots \exists y_k^j\psi[y_j/x_i, y_{i+1}^j/x_{i+1}, y_{i+2}^j/x_{i+2}, \ldots, y_k^j/x_k]\Big). \tag{3}$$

This reduction is an FPT-reduction, since the size of formula φ_2 is a function of the size of formula φ_1. Let $c_1, \ldots, c_{i-1}$ be arbitrary but fixed elements of the universe A of $\mathcal{A}$. We will show that $\varphi_1(x_1, \ldots, x_{i-1}) \equiv \varphi_2(x_1, \ldots, x_{i-1})$ by proving (a) $\varphi_1(c_1, \ldots, c_{i-1}) \to \varphi_2(c_1, \ldots, c_{i-1})$ and (b) $\varphi_2(c_1, \ldots, c_{i-1}) \to \varphi_1(c_1, \ldots, c_{i-1})$. For (a) assume that $\varphi_1(c_1, \ldots, c_{i-1})$ is true. This means, $\varphi_1[c_i/x_i]$ is true for all $c_i \in A$, that is for all $c_i \in A$ there exists an assignment to $x_{i+1}, \ldots, x_k$ such that ψ is true. Part (1) of $\varphi_2(c_1, \ldots, c_{i-1})$ asks for some $c_i \in A$ such that there exists an assignment to $x_{i+1}, \ldots, x_k$ such that ψ is true. Part (2) asks for the existence of an assignment to $x_{i+1}, \ldots, x_k$ such that ψ is true for each of the cases where x_i is one of the elements $c_1, \ldots, c_{i-1}$. Part (3) asks for the existence of an assignment to

$x_{i+1}, \ldots, x_k$ such that ψ is true for each of the cases where x_i is one of the elements that are assigned to $x_{i+1}, \ldots, x_k$ in the model of Part (1). All these are special cases of the universal quantification over x_i, hence $\varphi_2(c_1, \ldots, c_{i-1})$ is true.

For direction (b) assume towards a contradiction that $\varphi_1(c_1, \ldots, c_{i-1})$ is false and that $\varphi_2(c_1, \ldots, c_{i-1})$ is true. Since φ_1 is false, there exists $c_i \in A$ such that $\varphi_1[c_i/x_i]$ is false. We perform a case distinction on the value c_i. First let $c_i = c_j$ for some $j \in \{1, \ldots, i-1\}$. Then let $c_{i+1}, \ldots, c_k$ be the assignments to variables $y^j_{i+1}, \ldots, y^j_k$ in the model of φ_2. The jth conjunct of Part (2) of φ_2 states that ψ holds for $x_i = x_j$ using the assignment $c_{i+1}, \ldots, c_k$. Hence, assigning $c_{i+1}, \ldots, c_k$ to variables $x_{i+1}, \ldots, x_k$ in φ_1 is a model for $\varphi_1[c_i/x_i]$, which contradicts our assumption. As the next case, let $c_{i+1}, \ldots, c_k$ be the assignment to variables $y_{i+1}, \ldots, y_k$ in the model of φ_2 and let $c_i = c_j$ for some $j \in \{i+1, \ldots, k\}$. Let $c'_{i+1}, \ldots, c'_k$ be the assignments to variables $y^j_{i+1}, \ldots, y^j_k$ in the model of φ_2. The conjunct with index j of Part (3) of φ_2 states that ψ holds for $x_i = x_j = c_j$ using the assignment $c'_{i+1}, \ldots, c'_k$. Hence, assigning $c'_{i+1}, \ldots, c'_k$ to variables $x_{i+1}, \ldots, x_k$ in φ_1 is a model for $\varphi_1[c_i/x_i]$, which contradicts our assumption. For the last case, let c_i be one of the remaining values. Let $\ell_1, \ldots, \ell_m$ be all the literals in ψ that contain x_i. All of them are inequalities of the form $x_i \neq x_j$ for $j \neq i$. Let c'_i be the assignment to y_i in the model of φ_2. Let $\ell'_1, \ldots, \ell'_m$ be the literals in $\psi[y_i/x_i, y_{i+1}/x_{i+1}, \ldots, y_k/x_k]$ in Part (1) of φ_2 that correspond to $\ell_1, \ldots, \ell_m$. We have no knowledge about the truth value of these literals ℓ'_j with $1 \leq j \leq m$, but all of the literals ℓ_j in ψ evaluate to true when assigning $c_{i+1}, \ldots, c_k$ to the variables $x_{i+1}, \ldots, x_k$. Since ψ is in negation normal form and the literals $\ell_1, \ldots, \ell_m$ never occur in unnegated form, that is as equalities, changing the truth value of these literal from false to true will never result in changing the truth value of the whole formula from true to false. But since c'_i together with $c_{i+1}, \ldots, c_k$ is a model of Part (1) of φ_2, it holds that for all values of c_i that we consider in this case, that $\varphi_1[c_i/x_i]$ is true, which contradicts our assumption. This completes the case distinction and we have $\varphi_1(x_1, \ldots, x_{i-1}) \equiv \varphi_2(x_1, \ldots, x_2)$. ◀

6 Short Generalized Hex is W[1]-complete

Short Generalized Hex	
Instance:	Graph $G = (V, E)$, vertices $s, t \in V$, vertex sets $V_1, V_2 \subseteq V$ with $V_1 \cap V_2 = \emptyset$, and integer ℓ.
Parameter:	ℓ.
Problem:	Decide whether Player 1 has a winning strategy with at most ℓ moves in the generalized Hex game (G, s, t, V_1, V_2).

A generalized Hex game (G, s, t, V_1, V_2) is a positional game (V', E'), where the positions V' and the winning configurations E' are defined as follows. Set V' contains all vertices of G, that is $V' = V$. Set E' contains a set of vertices $\{v_1, \ldots, v_k\}$ if and only if $\{v_1, \ldots, v_k\} \cup \{s, t\}$ form an $s - t$ path in G. Additionally, vertices in V_1 and V_2 are already claimed by player 1 and player 2, respectively. Since the set of winning configurations of Short Generalized Hex is only defined implicitly, the input size of Short Generalized Hex can be exponential smaller than the number of winning configurations.

▶ **Theorem 2.** Short Generalized Hex *is* W[1]*-complete.*

Proof. Hardness is already known [4]. For membership, we reduce Short Generalized Hex to MC($\forall^{\neq}$-FO). Let $(G, s, t, V_1, V_2, \ell)$ be an instance of Short Generalized Hex, where $G = (V, E)$. Claimed vertices V_1 and V_2 can be preprocessed: (i) every $v \in V_1$ and its incident edges are removed from G and the neighbourhood of v is turned into a clique;

(ii) every $v \in V_2$ and its incident edges are removed from G. Hence, w.l.o.g. we assume that $V_1 = V_2 = \emptyset$. We construct an instance $(\mathcal{A}, \varphi)$ of MC($\forall^{\neq}$-FO) as follows. Let *EDGE* be a binary relation symbol and let S and T be unary relation symbols. Then $\mathcal{A}$ is the $\{EDGE, S, T\}$-structure $(V, EDGE^{\mathcal{A}}, S^{\mathcal{A}}, T^{\mathcal{A}})$ with $EDGE^{\mathcal{A}} := E$, $S^{\mathcal{A}} := \{s\}$, and $T^{\mathcal{A}} := \{t\}$. The $\forall^{\neq}$-FO-formula φ is defined as $\varphi = \exists s \exists t \exists x_1 \forall y_1 \exists x_2 \forall y_2 \dots \forall y_{\ell-1} \exists x_\ell \exists z_1 \exists z_2 \dots \exists z_\ell \psi$, with

$$\psi \equiv S(s) \wedge T(t) \wedge \Big(EDGE(s,t) \vee \bigvee_{i=1}^{\ell} \bigvee_{j=1}^{i} \Big(EDGE(s,z_1) \wedge EDGE(z_j,t) \wedge path_{i,j}(x_1,\dots,x_i,z_1,\dots,z_j) \wedge diff_i(x_1,y_1,\dots,y_{i-1},x_i) \Big) \Big),$$

$$path_{i,j}(x_1,\dots,x_i,z_1,\dots,z_j) \equiv \bigwedge_{h=1}^{j-1} EDGE(z_h,z_{h+1}) \wedge \bigwedge_{h=1}^{j} \bigvee_{k=1}^{i} z_h = x_k,$$

$$diff_i(x_1,y_1,\dots,x_{i-1},y_{i-1},x_i) \equiv \bigwedge_{1 \le j < k \le i} x_j \neq x_k \wedge \bigwedge_{1 \le j < k \le i} y_j \neq x_k.$$

The intuition of φ is the following. The variables x_i, y_i, and z_i represent the moves of Player 1, the moves of Player 2, and the ordered (s,t)-path induced by Player 1's moves, respectively. The variables s and t represent the vertices of the same name. Formula φ expresses that there is either a direct edge between s and t or a s-t path of length j was played. The main disjunctions ($\bigvee$) ensure that we consider wins that take up to ℓ moves, and build s-t path of length up to ℓ. Subformula $path_{i,j}$ will be true if and only if the z variables form a path using only values of the selected values for the x variables. Subformula $diff_i$ ensures that all x variables are pairwise distinct and they are distinct from all y variables with smaller index.

We have $|\varphi| = \mathcal{O}(\ell^4)$, so this is indeed an FPT-reduction and W[1]-membership follows. ◀

7 Short Maker-Breaker is W[1]-complete

Short Maker-Breaker
Instance: Hypergraph $G = (V, E)$, vertex sets $V_1, V_2 \subseteq V$ with $V_1 \cap V_2 = \emptyset$, and integer ℓ.
Parameter: ℓ.
Problem: Decide whether Player 1 has a winning strategy with at most ℓ if vertices V_1 and V_2 are already claimed by Player 1 and Player 2, respectively.

▶ **Theorem 4.** Short Maker-Breaker *is* W[1]*-complete.*

Proof. For showing membership, we reduce Short Maker-Breaker to MC($\forall^{\neq}$-FO). Let (G, V_1, V_2, ℓ) be an instance of Short Maker-Breaker, where $G = (V, E)$ is a hypergraph. Claimed vertices V_1 and V_2 can be preprocessed: (i) every $v \in V_1$ is removed from V and every hyperedge $e \in E$; (ii) every $v \in V_2$ is removed from V and every hyperedge $e \in E$ with $v \in e$ is removed from E. Hence, w.l.o.g. we assume that $V_1 = V_2 = \emptyset$. We construct an instance $(\mathcal{A}, \varphi)$ of MC($\forall^{\neq}$-FO) as follows. Let *IN* and *SIZE* be binary relation symbols. Then $\mathcal{A}$ is the $\{IN, SIZE\}$-structure $(V \cup E \cup \{1,\dots,|V|\}, IN^{\mathcal{A}}, SIZE^{\mathcal{A}})$ with $IN^{\mathcal{A}} := \{(x,e) \mid x \in V, e \in E, x \in e\}$ and $SIZE^{\mathcal{A}} := \{(e,i) \mid e \in E, |e| = i\}$. Hence, the universe of $\mathcal{A}$ consists of the vertices of G, an element for each hyperedge, and an element for some bounded number of integers. The $\forall^{\neq}$-FO-formula φ is defined as $\varphi \equiv \exists x_1 \forall y_1 \dots \forall y_{\ell-1} \exists x_\ell \exists e \exists z_1 \exists z_2 \dots \exists z_\ell \psi$, with

$$\psi \equiv \bigvee_{1 \le j \le i \le \ell} \Big(diff_i(x_1,y_1,\dots,x_i) \wedge SIZE(e,j) \wedge \bigwedge_{k=1}^{j} \bigvee_{h=1}^{i} z_k = x_h \wedge \bigwedge_{1 \le k < h \le j} z_k \neq z_h \wedge \bigwedge_{k=1}^{j} IN(z_k,e) \Big).$$

The subformula $\mathit{diff}_i(x_1, y_1, \ldots, x_i)$ refers to the subformula with same name used in the proof of Theorem 2. That is, it ensures that all x variables are pairwise distinct and that they are distinct from all y variables with smaller index. The intuition of φ is the following. The variables x_i and y_i represent the moves of Maker and the moves of Breaker, respectively. The variables z_i represent the vertices forming the winning configuration of Maker and e represents the hyperedge of this winning configuration. The first disjunction ensures that we consider wins that take up to ℓ moves. The second disjunction ensures that we consider winning configurations that consist of up to i vertices. After checking that e has the correct size ($\mathit{SIZE}(e, j)$), we encode that the values of the z variables are contained in the hyperedge represented by e and that these variables are pairwise disjoint and selected among the moves of Maker (the x variables).

We have $|\varphi| = \mathcal{O}(\ell^4)$, so this is indeed an FPT-reduction and W[1]-membership follows.

For hardness, we reduce k-Multicolored Clique to Short Maker-Breaker. The reduction is essentially the same as the reduction used for showing W[1]-hardness of Generalized Hex [4]. The crucial observation is that the construction of Bonnet et al. [4] contains only a polynomial number of possible $s-t$ paths. Hence, we can encode every such $s-t$-path as a unique hyperedge denoting a winning configuration in Short Maker-Breaker. ◀

8 Short Maker-Maker is AW[*]-complete

Short Maker-Maker
Instance: Hypergraph $G = (V, E)$, vertex sets $V_1, V_2 \subseteq V$ with $V_1 \cap V_2 = \emptyset$, and integer ℓ.
Parameter: ℓ.
Problem: Decide whether Player 1 has a winning strategy with at most ℓ if vertices V_1 and V_2 are already claimed by Player 1 and Player 2.

▶ **Theorem 5.** Short Maker-Maker *is* AW[*]*-complete.*

Proof. For membership, we reduce Short Maker-Maker to MC(FO). Let (G, V_1, V_2, ℓ) be an instance of Short Maker-Maker, where $G = (V, E)$ is a hypergraph. We construct an instance $(\mathcal{A}, \varphi)$ of MC(FO) as follows. Let $\mathit{V1}$, $\mathit{V2}$, and EDGE be unary relation symbols. Let IN be a binary relation symbol. Then $\mathcal{A}$ is the $\{\mathit{V1}, \mathit{V2}, \mathit{EDGE}, \mathit{IN}\}$-structure $(V \cup E, \mathit{V1}^{\mathcal{A}}, \mathit{V2}^{\mathcal{A}}, \mathit{EDGE}^{\mathcal{A}}, \mathit{IN}^{\mathcal{A}})$ with $\mathit{V1}^{\mathcal{A}} := V_1$, $\mathit{V2}^{\mathcal{A}} := V_2$, $\mathit{EDGE}^{\mathcal{A}} := E$, and $\mathit{IN}^{\mathcal{A}} := \{(x, e) \mid x \in V, e \in E, x \in e\}$. Hence, the universe of $\mathcal{A}$ consists of the vertices and the hyperedges of G. The FO-formula φ is defined as $\varphi \equiv \exists x_1 \forall y_1 \ldots \forall y_{\ell-1} \exists x_\ell \psi$, with

$$\psi \equiv \bigvee_{i=0}^{\ell} \mathit{legalP1}_i(x_1, y_1, \ldots, x_\ell) \wedge \Big(\neg \mathit{legalP2}_{i-1}(x_1, y_1, \ldots, x_\ell) \vee \big(\mathit{winP1}_i(x_1, y_1, \ldots, x_\ell) \wedge \neg \mathit{winP2}_{i-1}(x_1, y_1, \ldots, x_\ell)\big)\Big).$$

$$\mathit{winP1}_i(x_1, y_1, \ldots, x_\ell) \equiv \exists e \forall z \mathit{EDGE}(e) \wedge \Big(\neg \mathit{IN}(z, e) \vee \mathit{V1}(z) \vee \bigvee_{j=1}^{i} z = x_j\Big),$$

$$\mathit{winP2}_i(x_1, y_1, \ldots, x_\ell) \equiv \exists e \forall z \mathit{EDGE}(e) \wedge \Big(\neg \mathit{IN}(z, e) \vee \mathit{V2}(z) \vee \bigvee_{j=1}^{i} z = y_j\Big).$$

Variable x_j represent Player 1's jth move and variable y_j represent Player 2's jth move. The first disjunction represents the number of moves i that Player 1 needs to win the game. Formula $\mathit{legalP1}_i$ (see [3]) ensures that variables $(x_j)_{1 \le j \le i}$ represent legal moves of Player 1, that is vertices not contained in V_1 or V_2 or previously played vertices. Analogously, $\mathit{legalP2}_i$ ensures that variables $(y_j)_{1 \le j \le i}$ represent legal moves of Player 2. Formula $\mathit{winP1}_i$ ensures

that Player 1 has won within the i first moves, that is, it has completed a hyperedge with V_1 and variables up to x_i. Analogously, $winP2_i$ ensures that Player 2 has won within the i first moves. We have $|\varphi| = \mathcal{O}(\ell^3)$ and $|\mathcal{A}| = \mathcal{O}(|G|^2)$, so this is indeed an FPT-reduction and AW[*]-membership follows.

For hardness, we reduce from the AW[*]-complete problem Short Generalized Geography on bipartite graphs. The reduction is deferred to the long version [3]. ◀

9 Short Enforcer-Avoider is co-W[1]-complete

Short Enforcer-Avoider
Instance: Hypergraph $G = (V, E)$, vertex sets $V_1, V_2 \subseteq V$ with $V_1 \cap V_2 = \emptyset$, and integer ℓ.
Parameter: ℓ.
Problem: Decide whether Player 1 has a winning strategy with at most ℓ moves if vertices V_1 and V_2 are already claimed by Player 1 and Player 2, respectively.

▶ **Theorem 6.** Short Enforcer-Avoider *is* co-W[1]*-complete.*

Proof. We show that the co-problem of Short Enforcer-Avoider is W[1]-complete. The co-problem of Short Enforcer-Avoider is to decide whether for all strategies of Enforcer, there exists a strategy of Avoider such that during the first ℓ moves, Avoider does not claim a hyperedge. Again, vertices V_1 and V_2 are already claimed by Enforcer and Avoider, respectively. We prove W[1]-hardness by a parameterized reduction from Independent Set and W[1]-membership by reduction to MC($\forall^{\neq}$-FO).

In the W[1]-complete Independent Set problem [6], the input is a graph $G = (V, E)$ and an integer parameter k, and the question is whether G has an independent set of size k, i.e., a set of k pairwise non-adjacent vertices. We construct a positional game $G' = (V', E')$ by replacing each vertex of G by a clique of size $k + 1$. The vertex set V' has vertices $v(1), \ldots, v(k + 1)$ for each vertex $v \in V$, and hyperedges are $E' = \{\{v(i), v(j)\} : v \in V \text{ and } 1 \leq i < j \leq k + 1\} \cup \{\{u(i), v(j)\} : uv \in E \text{ and } 1 \leq i, j \leq k + 1\}$. We claim that G has an independent set of size k if and only if Avoider does not claim a hyperedge in the first k moves in the positional game G' starting from the empty position, that is $V_1 = V_2 = \emptyset$. For the forward direction, suppose $I = \{v_1, \ldots, v_k\}$ is an independent set of G of size k. Then, a winning strategy for Avoider is to claim an unclaimed vertex from $\{v_i(1), \ldots, v_i(k + 1)\}$ at round $i \in \{1, \ldots, k\}$. We note that Enforcer cannot claim all the vertices from $\{v_i(1), \ldots, v_i(k + 1)\}$, since there are not enough moves to do so, and Avoider does not complete a hyperedge with this strategy. On the other hand, suppose Avoider has a winning strategy in k moves. For an arbitrary play by Enforcer, let $\{v_1(i_1), \ldots, v_k(i_k)\}$ denote the vertices claimed by Player 1. Then, $v_i \neq v_j$ and $v_i v_j \notin E$ for any $1 \leq i < j \leq k$, since Player 1 would otherwise claim all the vertices of a hyperedge. Therefore, $\{v_1, \ldots, v_k\}$ is an independent set of G of size k.

For membership, we reduce to MC($\forall^{\neq}$-FO). Let (G, V_1, V_2, ℓ) be an instance of the co-problem of Short Enforcer-Avoider where $G = (V, E)$ is a hypergraph. First we do some preprocessing. We remove all vertices from G that are contained in V_2, that is the vertices already claimed by Avoider. If this results in an empty hyperedge, the instance is a no-instance. Otherwise, we remove all hyperedges that contain a vertex in V_1, that is the vertices already claimed by Enforcer, since Avoider will never lose via these edges anymore. Finally, we remove all vertices from G that are contained in V_1. Let $G = (V, E)$ now refer to the outcome of this preprocessing. By construction all vertices of G are unoccupied and some vertices might not be contained in any hyperedge. If G contains

less than 2ℓ vertices we can solve the problem via brute force in FPT time. Hence, in what follows we assume that there are at least 2ℓ unoccupied vertices in G. We construct an instance $(\mathcal{A}, \varphi)$ of MC($\forall^{\neq}$-FO) as follows. Let $EDGE_i$ be a i-ary relation symbol for $1 \leq i \leq \ell$. Then $\mathcal{A}$ is the $\{EDGE_1, \ldots, EDGE_\ell\}$-structure $(V, {EDGE_1}^{\mathcal{A}}, \ldots, {EDGE_\ell}^{\mathcal{A}})$ with ${EDGE_i}^{\mathcal{A}} := \{(v_1, \ldots, v_i) \mid e \in E, |e| = i, e = \{v_1, \ldots, v_\ell\}\}$, that is ${EDGE_i}^{\mathcal{A}}$ contains every permutation of all hyperedges of cardinality i. The $\forall^{\neq}$-FO-formula φ is defined as

$$\varphi \equiv \forall y_1 \exists x_1 \forall y_2 \exists x_2 \ldots \exists x_\ell \; \mathit{diff}_\ell(y_1, x_1, \ldots, x_\ell) \wedge \bigwedge_{1 \leq i \leq \ell} \bigwedge_{\{z_1, \ldots, z_i\} \subseteq \{x_1, \ldots, x_\ell\}} \neg EDGE_i(z_1, \ldots, z_i),$$

where $\mathit{diff}_i(y_1, x_1, \ldots, x_i) \equiv \bigwedge_{1 \leq j < k \leq i} x_j \neq x_k \wedge \bigwedge_{1 \leq j \leq k \leq i} y_j \neq x_k$.

Subformula $\mathit{diff}_i(y_1, x_1, \ldots, x_i)$ ensures that all x variables are pairwise distinct and they are distinct from all y variables with index less or equal theirs. The intuition of φ is the following. The variables x_i and y_i represent the moves of Avoider and the moves of Enforcer, respectively. Avoider wins if the x variables do not cover a whole hyperedge after ℓ moves. We only have to check hyperedges of size up to ℓ. Hence, for each cardinality $i \leq \ell$, we check for all subsets $z_1, \ldots, z_\ell$ of the x variables that they do not form a hyperedge. Formula φ does not pose any restrictions on the y variables, that is we do not force Enforcer to pick unoccupied vertices. We call a move by Enforcer that picks an already occupied vertex cheating. To prove correctness, we need to show that whenever Enforcer has a winning strategy σ_E that involves cheating, Enforcer also has a winning strategy σ'_E without cheating. We construct σ'_E as follows. We follow strategy σ_E while σ_E does not perform a cheating move. If the next move would be a cheating move, we play a random unoccupied vertex instead and keep track of this vertex in a new set V_r. The next time we need to select a move, we construct a board state s by removing all vertices in V_r from the picks of Enforcer and query strategy σ_E on this state s. If the answer is an unoccupied vertex, we perform this move normally. If instead the answer is a previously played vertex (which might be in V_r), we play a random unoccupied vertex instead and add it to V_r. Since σ_E was a winning strategy, so is σ'_E. Hence, formula φ does not need to check if the y variables correspond to unoccupied vertices. The construction can be done by an FPT algorithm since for each hyperedge $e \in E$ of cardinality i, we create $i! \leq \ell!$ entries in the $EDGE_i$ relation. We have $|\varphi| = \mathcal{O}(\ell^\ell)$, so this is indeed an FPT reduction and W[1]-membership follows. ◀

10 Conclusion

We have seen that the parameterized complexity of short positional games depends crucially on whether both players compete for achieving winning sets, or whether the game can be seen as one player aiming to achieve a winning set and the other player merely blocking the moves of the first player. Naturally, blocking moves correspond to inequalities in first-order logic, and our $\forall^{\neq}$-FO fragment of first-order logic therefore captures that the universal player can only block moves of the existential player. Our W[1]-completeness of MC($\forall^{\neq}$-FO) has been used several times in this paper, but our transformation of $\forall^{\neq}$-FO formulas into Σ_1 formulas may have other uses. As a concrete example related to positional games, Bonnet et al. [4] established that Short Hex is FPT by expressing the problem as a FO formula, and making use of Frick and Grohe's meta-theorem [11], similarly as we did in Section 4. This establishes that the problem is FPT but the running time is non-elementary in ℓ. However, we remark that their FO formula is actually a $\forall^{\neq}$-FO formula of size polynomial in ℓ. Our transformation gives an equivalent Σ_1 formula whose length is single-exponential in ℓ, and the meta-theorem of Grohe and Wöhrle [12] then gives a running time for solving Short Hex that is triply-exponential in ℓ.

Acknowledgments We thank anonymous reviewers for helpful comments and we thank Yijia Chen and Paul Hunter for bringing Grohe and Wöhrle's work [12] to our attention.

References

1. Karl R. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter intractability II (extended abstract). In *Proceedings of the 10th Annual Symposium on Theoretical Aspects of Computer Science (STACS 93)*, volume 665 of *Lecture Notes in Computer Science*, pages 374–385. Springer, 1993. doi:10.1007/3-540-56503-5_38.
2. Karl R. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter tractability and completeness IV: on completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995. doi:10.1016/0168-0072(94)00034-Z.
3. Édouard Bonnet, Serge Gaspers, Antonin Lambilliotte, Stefan Rümmele, and Abdallah Saffidine. The parameterized complexity of positional games. *CoRR*, abs/1704.08536, 2017. URL: https://arxiv.org/abs/1704.08536.
4. Édouard Bonnet, Florian Jamain, and Abdallah Saffidine. On the complexity of connection games. *Theoretical Computer Science*, 644:2–28, 2016. doi:10.1016/j.tcs.2016.06.033.
5. Jesper Makholm Byskov. Maker-Maker and Maker-Breaker games are PSPACE-complete. Technical Report RS-04-14, BRICS, Department of Computer Science, Aarhus University, 2004.
6. Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer-Verlag, New York, 1999.
7. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.
8. Shimon Even and Robert Endre Tarjan. A combinatorial problem which is complete in polynomial space. *Journal of the ACM*, 23(4):710–719, 1976. doi:10.1145/321978.321989.
9. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 1998.
10. Fedor V. Fomin and Dániel Marx. FPT suspects and tough customers: Open problems of downey and fellows. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond – Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 457–468. Springer, 2012. doi:10.1007/978-3-642-30891-8_19.
11. Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48(6):1184–1206, 2001.
12. Martin Grohe and Stefan Wöhrle. An existential locality theorem. *Annals of Pure and Applied Logic*, 129(1):131–148, 2004. doi:10.1016/j.apal.2004.01.005.
13. Alfred W. Hales and Robert I. Jewett. Regularity and positional games. *Transactions of the American Mathematical Society*, 106:222–229, 1963.
14. Ming Yu Hsieh and Shi-Chun Tsai. On the fairness and complexity of generalized k-in-a-row games. *Theoretical Computer Science*, 385(1):88–100, 2007. doi:10.1016/j.tcs.2007.05.031.
15. Stefan Reisch. Gobang ist PSPACE-vollständig. *Acta Informatica*, 13(1):59–66, 1980.
16. Stefan Reisch. Hex ist PSPACE-vollständig. *Acta Informatica*, 15:167–191, 1981. doi:10.1007/BF00288964.
17. Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978. doi:10.1016/0022-0000(78)90045-4.
18. Allan Scott. *On the Parameterized Complexity of Finding Short Winning Strategies in Combinatorial Games*. PhD thesis, University of Victoria, 2009.

19 Allan Scott and Ulrike Stege. Parameterized chess. In *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC 2008)*, volume 5018 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 2008. doi:10.1007/978-3-540-79723-4_17.

20 Allan Scott and Ulrike Stege. Parameterized pursuit-evasion games. *Theoretical Computer Science*, 411(43):3845–3858, 2010. doi:10.1016/j.tcs.2010.07.004.

21 Detlef Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 6(6):505–526, 1996.

Directed Hamiltonicity and Out-Branchings via Generalized Laplacians*†

Andreas Björklund[1], Petteri Kaski[2], and Ioannis Koutis[3]

1 **Department of Computer Science, Lund University, Lund, Sweden**
andreas.bjorklund@yahoo.se
2 **Department of Computer Science, Aalto University, Helsinki, Finland**
petteri.kaski@aalto.fi
3 **Department of Computer Science, University of Puerto Rico – Rio Piedras, San Juan, Puerto Rico**
ioannis.koutis@upr.edu

Abstract

We are motivated by a tantalizing open question in exact algorithms: can we detect whether an n-vertex directed graph G has a Hamiltonian cycle in time significantly less than 2^n?

We present new randomized algorithms that improve upon several previous works:

1. We show that for any constant $0 < \lambda < 1$ and prime p we can **count** the Hamiltonian cycles modulo $p^{\lfloor (1-\lambda)\frac{n}{3p} \rfloor}$ in expected time less than c^n for a constant $c < 2$ that depends only on p and λ. Such an algorithm was previously known only for the case of counting modulo *two* [Björklund and Husfeldt, FOCS 2013].
2. We show that we can detect a Hamiltonian cycle in $O^*(3^{n-\alpha(G)})$ time and **polynomial space**, where $\alpha(G)$ is the size of the maximum independent set in G. In particular, this yields an $O^*(3^{n/2})$ time algorithm for bipartite directed graphs, which is faster than the exponential-space algorithm in [Cygan *et al.*, STOC 2013].

Our algorithms are based on the algebraic combinatorics of "incidence assignments" that we can capture through evaluation of determinants of Laplacian-like matrices, inspired by the Matrix–Tree Theorem for directed graphs. In addition to the novel algorithms for directed Hamiltonicity, we use the Matrix–Tree Theorem to derive simple algebraic algorithms for detecting out-branchings. Specifically, we give an $O^*(2^k)$-time randomized algorithm for detecting out-branchings with at least k internal vertices, improving upon the algorithms of [Zehavi, ESA 2015] and [Björklund *et al.*, ICALP 2015]. We also present an algebraic algorithm for the directed k-Leaf problem, based on a non-standard monomial detection problem.

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

Keywords and phrases counting, directed Hamiltonicity, graph Laplacian, independent set, k-internal out-branching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.91

* A full version of the paper is available at http://arxiv.org/abs/1607.04002.

† The research leading to these results has received funding from the Swedish Research Council grants VR 2012-4730 "Exact Exponential-Time Algorithms" and VR 2016-03855 "Algebraic Graph Algorithms" (A.B.), the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement 338077 "Theory and Practice of Advanced Search and Enumeration" (P.K.), and grant NSF CAREER CCF-1149048 (I.K.). Work done in part while the authors were at Dagstuhl Seminar 17041 in January 2017 and at the Simons Institute for the Theory of Computing in December 2015.

1 Introduction

The Hamiltonian cycle problem has played a prominent role in development of techniques for the design of *exact algorithms* for hard problems. The early $O^*(2^n)$ algorithms based on dynamic programming and inclusion-exclusion [1, 20, 19], remained un-challenged for several decades. In 2010, Björklund [3], gave a randomized algorithm running in $O(1.657^n)$ time for the case of undirected graphs. The algorithm taps into the power of algebraic combinatorics, and in particular determinants that enumerate cycle covers.

Despite this progress in the undirected Hamiltonian cycle problem, a substantial improvement in the more general directed version of the problem remains an open problem and a key challenge in the area of exact algorithms. The currently best known general algorithm runs in $O^*(2^{n-\Theta(\sqrt{n/\log n})})$ time [4], and there are no known connections with the theory of SETH-hardness [18] that would – at least partly – dash the hope for a faster algorithm.

A number of recent works have attempted to crack directed Hamiltonicity, revealing that the problem is indeed easier in certain restricted settings. Cygan and Pilipczuk [13] showed that the problem admits an $O^*(2^{(1-\epsilon_d)n})$ time algorithm for graphs with average degree bounded by d, where ϵ_d is a constant with a doubly exponential dependence on d. Cygan *et al.* [12] showed that the problem admits an $O^*(1.888^n)$ time randomized algorithm for bipartite graphs and that the parity of directed Hamiltonian cycles can also be computed within the same time bound. Björklund and Husfeldt [6] showed that the parity of Hamiltonian cycles can be computed in $O^*(1.619^n)$ randomized time in general directed graphs. Finally, Björklund *et al.* [5] showed that the problem can be solved in $O^*((2-\Theta(1))^n)$ time when the graph contains less than 1.038^n Hamiltonian cycles, via a reduction to the parity problem. In this paper we improve or generalize all of these works.

Our results. As one would expect, all recent "below-2^n" algorithm designs for the Hamiltonicity problem rely on algebraic combinatorics and involve formulas that enumerate Hamiltonian cycles. But somewhat surprisingly, none of these approaches employs the directed version of the Matrix–Tree Theorem (see e.g. Gessel and Stanley [17, §11]), one of the most striking and beautiful results in algebraic graph theory. The theorem enables the enumeration of spanning out-branchings, that is, rooted spanning trees with all arcs oriented away from the root, via a determinant polynomial. Our results in this paper derive from a detailed combinatorial understanding and generalization of this classical setup.

The combinatorial protagonist of this paper is the following notion that enables a "two-way" possibility to view each arc in a directed graph:

▶ **Definition 1** (Incidence assignment)**.** Let G be a directed graph with vertex set V and arc set E. For a subset $W \subseteq V$ we say that a mapping $\mu : W \to E$ is an *incidence assignment* if for all $u \in W$ it holds that $\mu(u)$ is incident with u.

In particular, looking at a single arc $uv \in E$, an incidence assignment μ can assign uv in two[1] possible ways: as an out-arc $\mu(u) = uv$ at u, or as an in-arc $\mu(v) = uv$ at v.

From an enumeration perspective the serendipity of this "two-way" possibility to assign an arc becomes apparent when one considers how an incidence assignment μ can realize a

[1] Strictly speaking we are here assuming that both $u \in W$ and $v \in W$. To break symmetry in our applications we do allow also situations where uv has only one possible assignment due to either $u \notin W$ or $v \notin W$.

directed cycle in its image $\mu(W)$. Indeed, let

$$u_1u_2,\ u_2u_3,\ \ldots,\ u_{\ell-1}u_\ell,\ u_\ell u_1 \in E$$

be the arcs of a directed cycle C of length $\ell \geq 2$ in G with $V(C) \subseteq W$. It is immediate that there are now exactly two[2] ways to realize C in the image $\mu(W)$. Namely, we can realize C either (i) using only in-arcs with

$$\mu(u_1) = u_\ell u_1, \quad \mu(u_2) = u_1u_2, \quad \mu(u_3) = u_2u_3, \quad \ldots, \quad \mu(u_\ell) = u_{\ell-1}u_\ell\,, \tag{1}$$

or (ii) using only out-arcs with

$$\mu(u_1) = u_1u_2, \quad \mu(u_2) = u_2u_3, \quad \mu(u_3) = u_3u_4, \quad \ldots, \quad \mu(u_\ell) = u_\ell u_1\,. \tag{2}$$

Incidence assignments thus enable two distinct ways to realize a *directed* cycle. Furthermore, it is possible to *switch* between (1) and (2) so that only the images of $u_1, u_2, \ldots, u_\ell$ under μ are affected. The algebraization of this combinatorial observation is at the heart of the directed Matrix–Tree Theorem (which we will review for convenience of exposition in Sect. 2) and all of our results in this paper.

Our warmup result involves a generalization of the directed Hamiltonian path problem, namely the *k-Internal Out-Branching* problem, where the goal is to detect whether a given directed graph contains a spanning out-branching that has at least k internal vertices. This is a well-studied problem on its own, with several successive improvements the latest of which is an $O^*(3.617^k)$ algorithm by Zehavi [24] and an $O^*(3.455^k)$ algorithm by Björklund *et al.* [9] for the undirected version of the problem.

Using a combination of the directed Matrix–Tree Theorem and a monomial-sieving idea due to Floderus *et al.* [15], in Sect. 3 we show the following:

▶ **Theorem 2** (Detecting a k-Internal Out-Branching)**.** *There exists a randomized algorithm that solves the k-internal out-branching problem in time $O^*(2^k)$ and with negligible probability of reporting a false negative.*

In the full version of the paper [10] we give a further application for the k-Leaf problem, that is, detecting a spanning out-branching with at least k leaves. We note that Gabizon *et al.* [16] have recently given another application of the directed Matrix–Tree Theorem for the problem of detecting out-branchings of bounded degree.

Proceeding to our two main results, in Sect. 4 we observe that the directed Matrix–Tree Theorem leads to a formula for computing the number of Hamiltonian paths in arbitrary characteristic by using a standard inclusion–exclusion approach, which leads to a formula that involves the summation of 2^n determinants. To obtain a below-2^n design, we present a way to randomize the underlying Laplacian matrix so that the number of Hamiltonian paths does not change but in expectation most of the summands vanish modulo a prime power. Furthermore, to efficiently list the non-vanishing terms, we use a variation of an algorithm of Björklund *et al.* [8] that was used for a related problem, computing the permanent modulo a prime power. This leads to our first main result:

▶ **Theorem 3** (Counting directed Hamiltonian cycles modulo a prime power)**.** *For all $0 < \lambda < 1$ there exists a randomized algorithm that, given an n-vertex directed graph and a prime p as input, counts the number of Hamiltonian cycles modulo $p^{\lfloor(1-\lambda)n/(3p)\rfloor}$ in expected time $O^*\big(2^{n(1-\lambda^2/(19p\log_2 p))}\big)$. The algorithm uses exponential space.*

[2] Again strictly speaking it will be serendipitous to break symmetry so that certain cycles will have only one realization instead of two.

A corollary of Theorem 3 is that if G has at most d^n Hamiltonian cycles, we can detect one in time $O(c_d^n)$, where d is any fixed constant and $c_d < 2$ is a constant that only depends on d. As a further corollary we obtain a randomized algorithm for counting Hamiltonian cycles in graphs of bounded average (out-)degree d in $O(2^{(1-\epsilon_d)n})$ time. The constant ϵ_d has a polynomial dependency in d. Previous algorithms had a constant ϵ_d with an exponential dependency on d [7, 13]. (The proofs of these results are relegated to the full version of the paper [10].)

Returning to undirected Hamiltonicity, a key to the algorithm in [3] was the observation that determinants enumerate all non-trivial cycle covers an even number of times. This is due to the fact that each undirected cycle can be traversed in both directions. By picking a special vertex, one can break symmetry and force this to happen only for non-Hamiltonian cycle covers, so that the corresponding monomials cancel in characteristic 2. In Sect. 5 we present a "quasi-Laplacian" matrix whose determinant enables a similar approach for the directed case via algebraic combinatorics of incidence assignments, and furthermore enables one to accommodate a speedup assuming the existence of a good-sized independent set. We specifically prove the following as our second main result:

▶ **Theorem 4** (Detecting a directed Hamiltonian cycle)**.** *There exists a randomized algorithm that solves the directed Hamiltonian cycle problem on a given directed graph G with a maximum independent set of size $\alpha(G)$, in $O^*(3^{n(G)-\alpha(G)})$ time, polynomial space and with negligible probability of reporting a false negative.*

Theorem 4 improves and generalizes the exponential-space algorithm of Cygan *et al.* [12].

Terminology and conventions. All graphs in this paper are directed and without loops and parallel arcs unless indicated otherwise. For an arc e starting from vertex u and ending at vertex v we say that u is the *tail* of e and v is the *head* of e. The vertices u and v are the *ends* of e. A directed graph is *connected* if the undirected graph obtained by removing orientation from the arcs is connected. A subgraph of a graph is *spanning* if the subgraph has the same set of vertices as the graph. A connected directed graph is an *out-branching* if every vertex has in-degree 1 except for the *root* vertex that has in-degree 0. We say that a vertex is *internal* to an out-branching if it has out-degree at least 1; otherwise the vertex is a *leaf* of the out-branching. The (*directed*) *Hamiltonian cycle* problem asks, given a directed graph G as input, whether G has a spanning directed cycle as a subgraph. The notation $O^*(\ )$ suppresses a multiplicative factor polynomial in the input size. We say that an event parameterized by n has *negligible* probability if the probability of the event tends to zero as n grows without bound.

2 The symbolic Laplacian of a directed graph

This section develops the relevant preliminaries on directed graph Laplacians.

Permutations and the determinant. A bijection $\sigma : U \to U$ of a finite set U is called a *permutation* of U. A permutation σ *moves* an element $u \in U$ if $\sigma(u) \neq u$; otherwise σ *fixes* u. The *identity* permutation fixes every element of U. A permutation σ of U is a *cycle* of *length* $k \geq 2$ if there exist distinct $u_1, u_2, \ldots, u_k \in U$ with $\sigma(u_1) = u_2$, $\sigma(u_2) = u_3, \ldots, \sigma(u_{k-1}) = u_k$, $\sigma(u_k) = u_1$ and σ fixes all other elements of U. Two cycles are *disjoint* if every point moved by one is fixed by the other. The set of all permutations of U forms the *symmetric group* $\mathrm{Sym}(U)$ with the composition of mappings as the product operation of the group.

Every nonidentity permutation factors into a unique product of pairwise disjoint cycles. The *sign* of a permutation σ that factors into c disjoint cycles of lengths $k_1, k_2, \ldots, k_c$ is $\operatorname{sgn}(\sigma) = (-1)^{\sum_{j=1}^{c}(k_j-1)}$. The sign of the identity permutation is 1.

The *determinant* of a square matrix A with rows and columns indexed by U is the multivariate polynomial

$$\det A = \sum_{\sigma \in \mathrm{Sym}(U)} \operatorname{sgn}(\sigma) \prod_{u \in U} a_{u,\sigma(u)} \,.$$

The punctured Laplacian determinant via incidence assignments. Let G be a directed graph with n vertices. Associate with each arc $uv \in E = E(G)$ an indeterminate x_{uv}. The *symbolic Laplacian* $L = L(G)$ of G is the $n \times n$ matrix with rows and columns indexed by the vertices $u, v \in V = V(G)$ and the (u, v)-entry defined[3] by

$$\ell_{uv} = \begin{cases} \sum_{w \in V : wu \in E} x_{wu} & \text{if } u = v; \\ -x_{uv} & \text{if } uv \in E; \\ 0 & \text{if } u \neq v \text{ and } uv \notin E. \end{cases} \tag{3}$$

Observe that for each $v \in V$ we have that column v of L sums to zero because the diagonal entries cancel the negative off-diagonal entries. Furthermore, for each $u \in V$ we have that the monomials on row u of L correspond to the arcs incident to u. Indeed, each monomial at the diagonal corresponds to an in-arc to u, and each monomial at an off-diagonal entry corresponds to an out-arc from u. Thus, selecting one monomial from each row corresponds to selecting an incidence assignment.

To break symmetry, select an $r \in V$. The symbolic Laplacian of G *punctured at* r is obtained from L by deleting both row r and column r. We write $L_r = L_r(G)$ for the symbolic Laplacian of G punctured at r. Let us write $\mathcal{B}_r = \mathcal{B}_r(G)$ for the set of all spanning out-branchings of G with root $r \in V$. The following theorem is well-known (see e.g. Gessel and Stanley [17, §11]) and is presented here for purposes of displaying a proof that presents the cancellation argument using incidence assignments.

▶ **Theorem 5** (Directed Matrix–Tree Theorem). $\det L_r = \sum_{H \in \mathcal{B}_r} \prod_{uv \in E(H)} x_{uv}$.

Proof. Let us abbreviate $V_r = V(G) \setminus \{r\}$ and study the determinant

$$\det L_r = \sum_{\sigma \in \mathrm{Sym}(V_r)} \operatorname{sgn}(\sigma) \prod_{u \in V_r} \ell_{u,\sigma(u)} \,. \tag{4}$$

In particular, let us fix an arbitrary permutation $\sigma \in \mathrm{Sym}(V_r)$ and study the monomials of the polynomial $\prod_{u \in V_r} \ell_{u,\sigma(u)}$ with the assumption that this polynomial is nonzero. From (3) it is immediate for each $u \in V_r$ that $\ell_{u,\sigma(u)}$ expands either (i) to the diagonal sum $\sum_{w \in V : wu \in E} x_{wu}$, which happens precisely when σ fixes u with $\sigma(u) = u$, or (ii) to the off-diagonal $-x_{uv}$, which happens precisely when σ moves u with $\sigma(u) = v$.

Let us write $M(\sigma)$ for the set of all incidence assignments $\mu : V_r \to E$ with the properties that (i) each $u \in V_r$ fixed by σ is assigned to an in-arc $\mu(u) = wu \in E$ for some $w \in V$, and (ii) each $u \in V_r$ moved by σ is assigned to the unique out-arc $\mu(u) = uv \in E$ with $\sigma(u) = v$.

[3] Recall that we assume that G is loopless so the entries with $u = v$ are well-defined.

Let us write $f = f(\sigma)$ for the number of elements in V_r fixed by σ. It is immediate by (i) and (ii) that we have

$$\prod_{u \in V_r} \ell_{u,\sigma(u)} = \sum_{\mu \in M(\sigma)} (-1)^{n-1-f(\sigma)} \prod_{u \in V_r} x_{\mu(u)}\,. \tag{5}$$

Next observe that from μ we can reconstruct $\sigma = \sigma(\mu)$ by (i) setting $\sigma(u) = u$ for each u assigned to an in-arc in μ, and (ii) setting $\sigma(u) = v$ for each u assigned to an out-arc uv in μ. Thus the union $M = \bigcup_{\sigma \in Sym(V_r)} M(\sigma)$ is disjoint. Let us call the elements of M *proper* incidence assignments. By (4) and (5) we have

$$\det L_r = \sum_{\mu \in M} (-1)^{n-1-f(\sigma)} \mathrm{sgn}(\sigma(\mu)) \prod_{u \in V_r} x_{\mu(u)}\,. \tag{6}$$

We claim that an incidence assignment μ is proper if and only if for every $u \in V_r$ there is exactly one $u' \in V_r$ such that $\mu(u')$ is an in-arc to u. For the "only if" direction, let σ be the permutation underlying a proper μ, and observe that vertices moved by σ partition to cycles so a σ never moves a vertex to a fixed vertex. Thus, we have $u' = u$ for the points fixed by σ, and $u' = \sigma^{-1}(u)$ is the vertex preceding u along a cycle of σ for points moved by σ. For the "if" direction, define $\sigma(u) = u$ if $u = u'$ and $\sigma(u') = u$ if $u' \neq u$. In the latter case we have $\mu(u') = u'u$, which means that $u'' \neq u'$ and thus $\sigma(u'') = u'$; by uniqueness of u' eventually a cycle must close so σ is a well-defined permutation underlying μ and thus μ is proper.

Let us write $\mathcal{H}_r$ for the set of all spanning subgraphs of G with the property that every vertex in V_r has in-degree 1 and the root r has in-degree 0. From the previous claim it follows that we can view the set $\mu(V_r) = \{\mu(u) : u \in V_r\}$ for a proper μ as an element of $\mathcal{H}_r$. Furthermore, $\mu(V_r)$ is connected (and hence a spanning out-branching with root r) if and only if $\mu(V_r)$ is acyclic.

Consider an arbitrary $H \in \mathcal{H}_r$. If H has a cycle, let C be the least cycle in H according to some fixed but arbitrary ordering of the vertices of G. (Observe that any two cycles in H must be vertex-disjoint and cannot traverse r because r has in-degree 0.) Now consider an arbitrary proper μ that realizes H by $\mu(V_r) = H$. The cycle C is realized in μ by either (1) (in which case $\sigma(\mu)$ fixes all vertices in C), or (2) (in which case $\sigma(\mu)$ traces the cycle C). Furthermore, we may switch between realizations (1) and (2) so that the number of fixed points in the underlying permutation changes by $|V(C)|$ and the sign of the underlying permutation gets multiplied by $(-1)^{|V(C)|-1}$. It follows that the realizations (1) and (2) have different signs and thus cancel each other in (6). If H does not have a cycle, that is, $H \in \mathcal{B}_r$, it follows that there is a unique proper μ that realizes H. Indeed, first observe that H can be realized only by assigning in-arcs since any assignment of an out-arc in μ implies a cycle in $H = \mu(V_r)$, a contradiction. Second, the in-arcs are unique since each $u \in V_r$ has in-degree 1 in H. Finally, since μ assigns only in-arcs the underlying permutation $\sigma(\mu)$ is the identity permutation which has $\mathrm{sgn}(\sigma(\mu)) = 1$ and $(-1)^{n-1-f(\sigma(\mu))} = 1$. Thus, each acyclic H contributes to (6) through a single $\mu \in M$ with coefficient 1. The theorem follows. ◀

3 Corollary for *k*-internal out-branchings

This section proves Theorem 2. We rely on a substitution idea of Floderus *et al.* [15, Theorem 1] to detect monomials with at least k distinct variables.

Let G be an n-vertex directed graph given as input together with a nonnegative integer k. Without loss of generality we may assume that $k \leq n - 1$. Iterate over all choices for a root vertex $r \in V$. Introduce an indeterminate y_u for each vertex $u \in V$ and an

indeterminate z_{uv} for each arc $uv \in E$. Introduce one further indeterminate t. Construct the symbolic Laplacian L of G given by (3) and with the assignment $x_{uv} = (1 + ty_u)z_{uv}$ to the indeterminate x_{uv} for each $uv \in E$. Puncture L at r to obtain L_r. Using, for example, Berkowitz's determinant circuit design [2] for an arbitrary commutative ring with unity, in time $O^*(1)$ build an arithmetic circuit $\mathcal{C}$ of size $O^*(1)$ for $\det L_r$. Viewing $\det L_r$ as a multivariate polynomial over the polynomial ring $R[t, y_u, z_{uv} : u \in V, uv \in E]$ where R is an abstract ring with unity, from Theorem 5 it follows that G has a spanning out-branching rooted at r with at least k internal vertices if and only if the coefficient of t^k in $\det L_r$ (which is a polynomial that is either identically zero or both (i) homogeneous of degree k in the indeterminates y_u and (ii) homogeneous of degree $n-1$ in the indeterminates z_{uv}) has a monomial that is multilinear of degree k in the indeterminates y_u. Indeed, observe that the substitution $x_{uv} = (1 + ty_u)z_{uv}$ tracks in the degree of the indeterminate y_u whether u occurs as an internal vertex or not; the indeterminates z_{uv} make sure that distinct spanning out-branchings will not cancel each other.

To detect a multilinear monomial in $\mathcal{C}$ restricted to the coefficient of t^k we can invoke [11, Lemma 1] or [21, Lemma 2.8]. This results in a randomized algorithm that runs in time $O^*(2^k)$ and has a negligible probability of reporting a false negative. This completes the proof of Theorem 2. ◀

4 Modular counting of Hamiltonian cycles

This section proves Theorem 3. Fix an arbitrary constant $0 < \lambda < 1$. Let $0 < \beta < 1/2$ be a constant whose precise value is fixed later. Let p be a prime and let G be an n-vertex directed graph with vertex set V and arc set E given as input. Without loss of generality (by splitting any vertex u into two vertices, s and t, with s receiving the out-arcs from u, and t receiving the in-arcs to u) we may count the spanning paths starting from s and ending at t instead of spanning cycles. Similarly, without loss of generality we may assume that $2 \leq p < n$. (Indeed, for $p \geq n$ the counting outcome from Theorem 3 is trivial.)

Sieving for Hamiltonian paths among out-branchings. Let $s, t \in V$ be distinct vertices. Let us write $\mathrm{hp}(G, s, t)$ for the set of spanning directed paths that start at s and end at t in G. Recall that we write $V_t = V \setminus \{t\}$ for the t-punctured version of the vertex set V. Let us also write $V_{st} = V \setminus \{s, t\}$. For $O \subseteq V_t$, let L_s^O be the matrix obtained from the Laplacian (3) by first puncturing at s and then substituting $x_{uv} = 0$ for all arcs $uv \in E$ with $u \in V_t \setminus O$. Since a path $P \in \mathrm{hp}(G, s, t)$ is precisely a spanning out-branching rooted at s such that every vertex $u \in V_t$ has out-degree 1, we have, by Theorem 5 and the principle of inclusion and exclusion,

$$\sum_{P \in \mathrm{hp}(G,s,t)} \prod_{uv \in E(P)} x_{uv} = \sum_{O \subseteq V_t} (-1)^{|V_t \setminus O|} \det L_s^O \,. \tag{7}$$

In particular observe that (7) holds in any characteristic.

Cancellation modulo a power of p. With foresight, select $k = \lfloor (1-\lambda)(1/2-\beta)n/p \rfloor$. Our objective is next to show that by carefully injecting entropy into the underlying Laplacian we can, in expectation and working modulo p^k, cancel all but an exponentially negligible fraction of the summands on the right-hand side of (7). Furthermore, we can algorithmically narrow down to the nonzero terms, leading to an exponential improvement to 2^n.

Let us assign $x_{uv} = 1$ for all $uv \in E$ with $u \neq t$. Since no spanning path that ends at t may contain an arc $tu \in E$ for any $u \in V_t$, we may without loss of generality assume that G contains

all such arcs, and assign, independently and uniformly at random $x_{tu} \in \{0, 1, \ldots, p-1\}$. Thus, the summands $\det L_s^O$ for $O \subseteq V_t$ are now integer-valued random variables and (7) evaluates to $|\mathrm{hp}(G, s, t)|$ with probability 1.

Let us next study a fixed $O \subseteq V_t$. Let F_O be the event that L_s^O has no more than k rows where each entry is divisible by p. In particular, $\det L_s^O \not\equiv 0 \pmod{p^k}$ implies F_O. To bound the probability of F_O from above, observe that L_s^O is identically zero at each row $u \in V_{st} \setminus O$ except possibly at the diagonal entries. Furthermore, because of the random assignment to the indeterminates x_{tu}, each diagonal entry at these rows is divisible by p with probability $1/p$. Let us take this intuition and turn it into a listing algorithm for (a superset of the) sets $O \subseteq V_t$ that satisfy F_O.

Bipartitioning. For listing we will employ a meet-in-the-middle approach based on building each set $O \subseteq V_t$ from two parts using the following bipartitioning. Let $V_t^{(1)} \cup V_t^{(2)} = V_t$ be a bipartition with $|V_t^{(1)}| = \lceil n/3 \rceil$ and $|V_t^{(2)}| = n - 1 - \lceil n/3 \rceil$. Associate with each $O_1 \subseteq V_t^{(1)}$ a vector $z^{O_1} \in \{0, 1, \ldots, p-1, \infty\}^{V_{st}}$ with the entry at $u \in V_{st}$ defined by

$$z_u^{O_1} = \begin{cases} \infty & \text{if } u \in O_1; \\ \left(x_{tu} + \sum_{w \in O_1 : wu \in E} x_{wu}\right) \bmod p & \text{otherwise.} \end{cases} \tag{8}$$

Similarly, associate with each $O_2 \subseteq V_t^{(1)}$ a vector $z^{O_2} \in \{0, 1, \ldots, p-1, \infty\}^{V_{st}}$ with the entry at $u \in V_{st}$ defined by

$$z_u^{O_2} = \begin{cases} \infty & \text{if } u \in O_2; \\ \left(-\sum_{w \in O_2 : wu \in E} x_{wu}\right) \bmod p & \text{otherwise.} \end{cases} \tag{9}$$

Suppose now that we have $O_1 \subseteq V_t^{(1)}$ and $O_2 \subseteq V_t^{(2)}$ with $O = O_1 \cup O_2$. We claim that F_O holds only if the vectors z^{O_1} and z^{O_2} agree in at most k entries. Indeed, observe that $z_u^{O_1} = z_u^{O_2}$ holds only if both $u \in V_{st} \setminus O$ and the (u, u)-entry of L_s^O is divisible by p. That is, $z_u^{O_1} = z_u^{O_2}$ implies the entire row u of L_s^O consists only of elements divisible by p. Thus it suffices to list all pairs (O_1, O_2) such that z^{O_1} and z^{O_2} have at most k agreements.

Balanced and unbalanced sets. To set up the listing procedure, let us now partition the index domain V_{st} of our vectors into $b = \lfloor 3 \log_2 p \rfloor$ pairwise disjoint sets $S_1, S_2, \ldots, S_b$ such that we have $\lfloor (n-2)/b \rfloor \leq |S_i| \leq \lceil (n-2)/b \rceil$.

Let us split the sets $O \subseteq V_t$ into two types. Let us say that O is *balanced* if $(1/2 - \beta)n/b \leq |(V_{st} \setminus O) \cap S_i| \leq (1/2 + \beta)n/b$ holds for all $i = 1, 2, \ldots, b$; otherwise O is *unbalanced*. Recalling that $\sum_{j=0}^{\ell} \binom{n}{j} \leq 2^{nH(\ell/n)}$ holds for all integers $1 \leq \ell \leq n/2$, where $H(\rho) = -\rho \log_2 \rho - (1-\rho)\log_2(1-\rho)$ is the binary entropy function, observe that there are in total at most

$$\begin{aligned} 2^{n+1-\min_i |S_i|} b \sum_{j=0}^{\lceil (1/2-\beta)n/b \rceil} \binom{\lfloor n/b+2 \rfloor}{j} &\leq 2^{n-(n-2)/b+3} 2^{(n/b+2)H(1/2-\beta)} b \\ &\leq 2^{n(1-(1-H(1/2-\beta))/b)+7} b \end{aligned} \tag{10}$$

sets O that are unbalanced.

Precomputation and listing. Suppose that $O_1 \subseteq V_t^{(1)}$ and $O_2 \subseteq V_t^{(1)}$ are *compatible* in the sense that z^{O_1} and z^{O_2} agree in at most k entries. For $S \subseteq V_{st}$ and a vector z whose entries

are indexed by V_{st}, let us write z_S for the restriction of z to S. If O_1 and O_2 are compatible, then by an averaging argument there must exist an $i = 1, 2, \ldots, b$ such that $z_{S_i}^{O_1}$ and $z_{S_i}^{O_2}$ agree in at most k/b entries. In particular, this enables us to iterate over O_2 and list all compatible O_1 by focusing only on each restriction to S_i for $i = 1, 2, \ldots, b$. Furthermore, the search inside S_i can be precomputed to look-up tables. Indeed, for each $i = 1, 2, \ldots, b$ and each key $g \in \{0, 1, \ldots, p-1, \infty\}^{S_i}$, let us build a complete list of all subsets $O_1 \subseteq V_t^{(1)}$ such that $z_{S_i}^{O_1}$ and g agree in at most k/b entries. These b look-up tables can be built by processing in total at most

$$\sum_{i=1}^{b} 2^{V_t^{(1)}} (p+1)^{|S_i|} \leq 2^{n/3+7} 2^{(n/(\lfloor 3\log_2 p \rfloor)+2)\log_2(p+1)} \log_2 p = O(2^{0.87n})$$

pairs (O_1, g). This takes time $O^*(2^{0.87n})$ in total.

The main listing procedure now considers each $O_2 \subseteq V_t^{(2)}$ in turn, and for each $i = 1, 2, \ldots, b$ consults the look-up table for direct access to all O_1 such that $z_{S_i}^{O_1}$ and $z_{S_i}^{O_2}$ agree in at most k/b entries. In particular this will list all compatible pairs (O_1, O_2) and hence all sets $O = O_1 \cup O_2$ such that F_O holds.

Expected running time. Let us now analyze the expected running time of the algorithm. We start by deriving an upper bound for the expected number of pairs (O_1, O_2) considered by the main listing procedure. First, observe that the total number of pairs (O_1, O_2) considered by the procedure with $O = O_1 \cup O_2$ unbalanced is bounded from above by our upper bound (10) for the total number of unbalanced O. Indeed, $O_1 = O \cap V_t^{(1)}$ and $O_2 = O \cap V_t^{(2)}$ are uniquely determined by O.

Next, for a pair (O_1, O_2) with balanced $O = O_1 \cup O_2$ and $i = 1, 2, \ldots, b$, let $G_{O_1,O_2,i}$ be the event that $z_{S_i}^{O_1}$ and $z_{S_i}^{O_2}$ agree in at most k/b entries. We seek an upper bound for the probability of $G_{O_1,O_2,i}$ to obtain an upper bound for the expected number of pairs with balanced $O = O_1 \cup O_2$ considered by the main listing procedure. Let $A_{O_1,O_2,i}$ be the number of entries in which $z_{S_i}^{O_1}$ and $z_{S_i}^{O_2}$ agree. We observe that $A_{O_1,O_2,i}$ is binomially distributed with expectation $|(V_t \setminus O) \cap S_i|/p$. Since O is balanced, we have $(1/2 - \beta)n/b \leq |(V_t \setminus O) \cap S_i| \leq (1/2 + \beta)n/b$. We also recall that $k = \lfloor (1-\lambda)(1/2-\beta)n/p \rfloor$. A standard Chernoff bound now gives

$$\begin{aligned}
\Pr(G_{O_1,O_2,i}) &\leq \Pr\big(A_{O_1,O_2,i} \leq k/b\big) \\
&\leq \Pr\big(A_{O_1,O_2,i} \leq (1-\lambda)|(V_t \setminus O) \cap S_i|/p\big) \\
&\leq \exp\big(-\lambda^2 |(V_t \setminus O) \cap S_i|/(2p)\big) \\
&\leq \exp\big(-\lambda^2 (1/2-\beta)n/(2pb)\big)\,.
\end{aligned}$$

Recalling that $b = \lfloor 3\log_2 p \rfloor$, the main listing procedure thus considers in expectation at most $2^n \exp\big(-\lambda^2(1/2-\beta)n/(2p(3\log_2 p))\big)$ pairs (O_1, O_2) with $O = O_1 \cup O_2$ balanced. Recalling our upper bound for the total number of unbalanced sets (10), we thus have that the main listing procedure runs in $O^*\big(2^n \exp\big(-\lambda^2(1/2-\beta)n/(2p(3\log_2 p))\big) + 2^{n(1-(1-H(1/2-\beta))/(3\log_2 p))}\big)$ expected time. Recalling that precomputation runs in $O^*(2^{0.87n})$ time, we thus have for $\beta = 1/6$ that the entire algorithm runs in $O^*(2^{n(1-\lambda^2/(19p\log_2 p))})$ expected time and computes $|\mathrm{hp}(G, s, t)|$ modulo $p^{\lfloor (1-\lambda)n/(3p) \rfloor}$. This completes the proof of Theorem 3. ◀

5 Directed Hamiltonicity via quasi-Laplacian determinants

This section proves Theorem 4. Let G be a directed n-vertex graph given as input.

Finding a maximum independent set. Let $B \cup Y = V(G)$ be a partition of the vertex set into two disjoint sets B ("blue") and Y ("yellow") such that no arc has both of its ends in Y. That is, Y is an independent set.

We can find an Y of the maximum possible size as follows. First, in time polynomial in n compute the maximum-size matching in the undirected graph obtained from G by disregarding the orientation of the arcs. This maximum-size matching must consist of at least $\lfloor n/2 \rfloor$ edges or G does not admit a Hamiltonian cycle. (Indeed, from a Hamiltonian cycle we can obtain a matching with $\lfloor n/2 \rfloor$ edges by taking every other arc in the cycle.) Since for each matching edge it holds that both ends of the edge cannot be in an independent set, we can in time $O^*(3^{n/2})$ find a maximum-size independent set Y of G. Furthermore, $\alpha(G) = |Y| \leq \lfloor n/2 \rfloor + 1$, so we are within our budget of $O^*(3^{n(G)-\alpha(G)})$ in terms of running time. In fact, $|Y| \leq \lfloor n/2 \rfloor$ or otherwise G trivially does not admit a Hamiltonian cycle.

The symbolic quasi-Laplacian. We will first define the quasi-Laplacian and then give intuition for its design. Let us work over a field of characteristic 2. For each $y \in Y$ introduce a copy y_{in} and let Y_{in} be the set of all such copies. Similarly, for each $y \in Y$ introduce a copy y_{out} and let Y_{out} be the set of all such copies. We assume that Y_{in} and Y_{out} are disjoint. For each $j \in Y_{\text{in}} \cup Y_{\text{out}}$ let us write $\underline{j} \in Y$ for the underlying element of Y of which j is a copy. Let B_* be a set of $n - 2|Y|$ elements that is disjoint from both Y_{in} and Y_{out}. For each $uv \in E$ and each $j \in B_* \cup Y_{\text{in}} \cup Y_{\text{out}}$, introduce an indeterminate $x_{uv}^{(j)}$.

Select an arbitrary vertex $s \in B$ for purposes of breaking symmetry and let $I, O \subseteq B$. The *quasi-Laplacian* $Q^{I,O,s} = Q^{I,O,s}(G)$ of G with *skew at* s be the $n \times n$ matrix whose rows are indexed by $u \in B \cup Y$ and whose columns are indexed by $j \in B_* \cup Y_{\text{in}} \cup Y_{\text{out}}$ such that the (u, j)-entry is defined by

$$q_{uj}^{I,O,s} = \begin{cases} \sum_{w \in O : wu \in E, u \in I} x_{wu}^{(j)} & \\ \quad + \sum_{w \in I : uw \in E, u \in O \setminus \{s\}} x_{uw}^{(j)} & \text{(a) if } u \in B \text{ and } j \in B_*; \\ x_{u\underline{j}}^{(j)} & \text{(b) if } u \in O \setminus \{s\} \text{ and } j \in Y_{\text{in}} \text{ with } u\underline{j} \in E; \\ x_{\underline{j}u}^{(j)} & \text{(c) if } u \in I \text{ and } j \in Y_{\text{out}} \text{ with } \underline{j}u \in E; \\ \sum_{w \in O : wu \in E} x_{wu}^{(j)} & \text{(d) if } u \in Y \text{ and } j \in Y_{\text{in}} \text{ with } u = \underline{j}; \\ \sum_{w \in I : uw \in E} x_{uw}^{(j)} & \text{(e) if } u \in Y \text{ and } j \in Y_{\text{out}} \text{ with } u = \underline{j}; \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Let us next give some intuition for (11) before proceeding with the proof.

Analogously to the Laplacian (3), the quasi-Laplacian (11) has been designed so that the monomials of each row $u \in B \cup Y$ of $Q^{I,O,s}$ control the assignment of either an in-arc or an out-arc to u in an incidence assignment, and the skew at s is used to break symmetry so that s is always assigned an in-arc to s. In particular, without the skew at row s and with $I = O$, each column of $Q^{I,O,s}$ would sum to zero, in analogy with the (non-punctured) Laplacian.

Let us now give intuition for the columns $j \in Y_{\text{in}}$ and $j \in Y_{\text{out}}$. First, observe by (b) and (d) in (11) that selecting a monomial from column $j \in Y_{\text{in}}$ corresponds to making sure that the in-degree of $\underline{j}$ is 1. Such a selection may be either a "quasi-diagonal" assignment of the in-arc $w\underline{j}$ to $u = \underline{j} \in Y$ via (d) for some $w \in B$; or an "off-diagonal" assignment of the out-arc $u\underline{j}$ to $u \in B$ via (b). Second, observe by (c) and (e) in (11) that selecting a monomial from column $j \in Y_{\text{out}}$ corresponds to making sure that the out-degree of $\underline{j}$ is 1. *Thus, the columns $j \in Y_{\text{in}} \cup Y_{\text{out}}$ enable us to make sure that an incidence assigment has both in-degree* 1 *and out-degree* 1 *at each $u \in Y$ without the use of sieving. This gives us the speed-up from $O^*(3^n)$ to $O^*(3^{n-|Y|})$ running time.* Observe also that the structure for the quasi-Laplacian

$Q^{I,O,s}$ is enabled precisely because Y is an independent set and thus no arc contributes to both in-degree and out-degree in Y.

The quasi-Laplacian determinant sieve. Recalling that we are working over a field of characteristic 2, let us study the sum

$$\sum_{\substack{I,O\subseteq B\\ I\cup O=B\\ s\in I}} \det Q^{I,O,s} = \sum_{I\subseteq B}\sum_{O\subseteq B} \det Q^{I,O,s} = \sum_{\substack{\sigma:B\cup Y\to B_*\cup Y_{\text{in}}\cup Y_{\text{out}}\\ \sigma \text{ bijective}}} \sum_{I\subseteq B}\sum_{O\subseteq B}\prod_{u\in B\cup Y} q^{I,O,s}_{u,\sigma(u)}\,. \quad (12)$$

Observe that the first equality in (12) holds because $Q^{I,O,s}$ has by (11) an identically zero row unless $I\cup O = B$ and $s\in I$; the second equality holds by definition of the determinant in characteristic 2 and changing the order of summation. From (11) and the right-hand side of (12) it is immediate that (12) is either identically zero or a homogeneous polynomial of degree n in the $n|E|$ indeterminates $x^{(j)}_{uv}$ for $j\in B_*\cup Y_{\text{in}}\cup Y_{\text{out}}$ and $uv\in E$. [4] We claim that (12) is not identically zero if and only if G admits at least one spanning cycle. Furthermore, each spanning cycle in G defines precisely $|B_*|!$ distinct monomials in (12).

To establish the claim, fix a bijection $\sigma : B\cup Y\to B_*\cup Y_{\text{in}}\cup Y_{\text{out}}$. Let us write $M(\sigma)$ for the set of all incidence assignments $\mu : B\cup Y\to E$ that are *proper* in the sense that all of the following six requirements hold (cf. (11)):

- (s): $\mu(s)$ is an in-arc to s;
- (a): for all $u\in B$ with $\sigma(u)\in B_*$ it holds that $\mu(u)$ has both of its vertices in B;
- (b): for all $u\in B$ with $\sigma(u)\in Y_{\text{in}}$ it holds that $\mu(u)$ is an in-arc to $\sigma(u)$;
- (c): for all $u\in B$ with $\sigma(u)\in Y_{\text{out}}$ it holds that $\mu(u)$ is an out-arc from $\sigma(u)$;
- (d): for all $u\in Y$ with $\sigma(u)\in Y_{\text{in}}$ it holds that $\mu(u)$ is an in-arc to u and $u=\sigma(u)$; and
- (e): for all $u\in Y$ with $\sigma(u)\in Y_{\text{out}}$ it holds that $\mu(u)$ is an out-arc from u and $u=\sigma(u)$.

Observe that each $\mu\in M(\sigma)$ defines a collection of n arcs $\mu(B\cup Y)=\{\mu(u) : u\in B\cup Y\}$. Let us write Z^μ_{in} (respectively, Z^μ_{out}) for the set of vertices in $B\cup Y$ with zero in-degree (out-degree) with respect to the arcs in $\mu(B\cup Y)$. Since σ is a bijection and thus has a preimage for each $j\in Y_{\text{in}}\cup Y_{\text{out}}$, from (b,c,d,e) above it follows that $Z^\mu_{\text{in}}\subseteq B$ and $Z^\mu_{\text{out}}\subseteq B$. Furthermore, from (a,b,c,d,e) it follows that for the arcs in $\mu(B\cup Y)$ the sum of the in-degrees (and the sum of the out-degrees) of the vertices in B is $|B| = |B_*|+|Y|$. Thus, we have that Z^μ_{in} and Z^μ_{out} are both empty if and only if for the arcs $\mu(B\cup Y)$ both the in-degree and the out-degree of every vertex $u\in B\cup Y$ is 1. (Note that the claim is immediate for $u\in Y$ by (b,c,d,e) and bijectivity of σ.)

Let us now study the right-hand side of (12) for a fixed σ. Using (a,b,c,d,e) and (11) to rearrange in terms of incidence assignments, we have

$$\sum_{I\subseteq V}\sum_{O\subseteq V}\prod_{u\in B\cup Y} q^{I,O,s}_{u,\sigma(u)} = \sum_{\mu\in M(\sigma)}\prod_{u\in B\cup Y} x^{(\sigma(u))}_{\mu(u)} \sum_{I\subseteq Z^\mu_{\text{in}}}\sum_{O\subseteq Z^\mu_{\text{out}}} 1\,. \quad (13)$$

Since we are working in characteristic 2, all other $\mu\in M(\sigma)$ except those for which $\mu(B\cup Y)$ is a cycle cover will cancel in the right-hand side of (13).

Take the sum of (13) over all bijections σ. Consider an arbitrary cycle cover of $B\cup Y$. Let C be a cycle in this cycle cover. Assuming that C does not contain s, we can realize C in an incidence assignment $\mu : B\cup Y\to E$ either using (1) or (2). If C contains s and μ is proper,

[4] Our algorithm for deciding Hamiltonicity will naturally not work with a symbolic representation of (12) but rather in a homomorphic image under a random evaluation homomorphism.

only the realization (1) is possible by (s). To see that realization with a proper $\mu \in M(\sigma)$ for some σ is possible, consider an arbitrary $u \in B \cup Y$ and observe that each image $\mu(u)$ by (b,c,d,e) uniquely determines the image $\sigma(u) \in Y_{\text{in}} \cup Y_{\text{out}}$ when $\mu(u)$ has one vertex in Y; when $\mu(u)$ has both vertices in B, an unused $\sigma(u) \in B_*$ may be chosen arbitrarily so that $\mu \in M(\sigma)$. It follows that any cycle cover with c cycles is realized as exacly $|B_*|!$ distinct monomials $\prod_{u \in B \cup Y} x_{\mu(u)}^{(\sigma(u))}$ in (12), each with coefficient 2^{c-1}. This coefficient is nonzero if and only if $c = 1$.

Completing the algorithm. To detect whether the given n-vertex directed graph G admits a Hamiltonian cycle, first decompose the vertex set into disjoint $V = B \cup Y$ with Y an independent set of size $|Y| = \alpha(G)$ using the algorithm described earlier. Next, in time $O^*(1)$ construct an irreducible polynomial of degree $2\lceil \log_2 n \rceil$ over $\mathbb{F}_2$ (see e.g. von zur Gathen and Gerhard [23, §14.9]) to enable arithmetic in the finite field of order $q = 2^{2\lceil \log_2 n \rceil} \geq n^2$ in time $O^*(1)$ for each arithmetic operation. Next, assign an independent uniform random value from $\mathbb{F}_q$ to each indeterminate $x_{uv}^{(j)}$ with $j \in B_* \cup Y_{\text{in}} \cup Y_{\text{out}}$ and $uv \in E$. Finally, using the assigned values for the indeterminates, compute the left-hand side of (12) using, for example, Gaussian elimination to compute each determinant $\det Q^{I,O,s}$ in $O^*(1)$ operations in $\mathbb{F}_q$. Let us write $r \in \mathbb{F}_q$ for the result of this computation. In particular, we can compute r from a given G in total $O^*(3^{|B|}) = O^*(3^{n(G)-\alpha(G)})$ operations in $\mathbb{F}_q$, and consequently in total $O^*(3^{n(G)-\alpha(G)})$ time. If (12) is identically zero, then clearly $r = 0$ with probability 1. If (12) is not identically zero (and hence a homogeneous polynomial of degree $d = n$ in the indeterminates) then by the DeMillo–Lipton–Schwartz–Zippel lemma [14, 22, 25] we have $r \neq 0$ with probability at least $1 - d/q \geq 1 - n/n^2 \geq 1 - 1/n = 1 - o(1)$. Thus we can decide whether G is Hamiltonian based on whether $r \neq 0$. In particular this gives probability $o(1)$ of reporting a false negative. This completes the proof of Theorem 4. ◀

References

1 Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, January 1962. doi:10.1145/321105.321111.

2 Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150, 1984. doi:10.1016/0020-0190(84)90018-8.

3 Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM Journal on Computing*, 43(1):280–299, 2014. doi:10.1137/110839229.

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

5 Andreas Björklund, Holger Dell, and Thore Husfeldt. The parity of set systems under random restrictions with applications to exponential time problems. In *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 231–242, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-47672-7_19.

6 Andreas Björklund and Thore Husfeldt. The parity of directed hamiltonian cycles. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 724–735, Oct 2013. doi:10.1109/FOCS.2013.83.

7 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The traveling salesman problem in bounded degree graphs. *ACM Trans. Algorithms*, 8(2):18:1–18:13, April 2012. doi:10.1145/2151171.2151181.

8 Andreas Björklund, Thore Husfeldt, and Isak Lyckberg. Computing the permanent modulo a prime power. Unpublished manuscript, 2015.

9 Andreas Björklund, Vikram Kamat, Lukasz Kowalik, and Meirav Zehavi. Spotting trees with few leaves. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 243–255. Springer, 2015. doi:10.1007/978-3-662-47672-7_20.

10 Andreas Björklund, Petteri Kaski, and Ioannis Koutis. Directed Hamiltonicity and Out-Branchings via Generalized Laplacians. *CoRR*, abs/1607.04002, 2017. URL: http://arxiv.org/abs/1607.04002.

11 Andreas Björklund, Petteri Kaski, and Łukasz Kowalik. Constrained multilinear detection and generalized graph motifs. *Algorithmica*, 74(2):947–967, 2016. doi:10.1007/s00453-015-9981-1.

12 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC'13, pages 301–310, New York, NY, USA, 2013. ACM. doi:10.1145/2488608.2488646.

13 Marek Cygan and Marcin Pilipczuk. Faster exponential-time algorithms in graphs of bounded average degree. *Information and Computation*, 243:75–85, 2015. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013). doi:10.1016/j.ic.2014.12.007.

14 R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7:193–195, 1978.

15 Peter Floderus, Andrzej Lingas, Mia Persson, and Dzmitry Sledneu. Detecting monomials with k distinct variables. *Information Processing Letters*, 115(2):82–86, 2015. doi:10.1016/j.ipl.2014.07.003.

16 Ariel Gabizon, Daniel Lokshtanov, and Michał Pilipczuk. Fast algorithms for parameterized problems with relaxed disjointness constraints. In *Algorithms – ESA 2015: 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 545–556, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-48350-3_46.

17 Ira M. Gessel and Richard P. Stanley. Algebraic enumeration. In R. L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume II, pages 1021–1061. North-Holland, Amsterdam, 1995.

18 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

19 Richard M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operations Research Letters*, 1(2):49–51, 1982. doi:10.1016/0167-6377(82)90044-X.

20 Samuel Kohn, Allan Gottlieb, and Meryle Kohn. A generating function approach to the traveling salesman problem. In *Proceedings of the Annual Conference (ACM'77), Association for Computing Machinery*, pages 294–300, 1977.

21 Ioannis Koutis and Ryan Williams. Limits and applications of group algebras for parameterized problems. *ACM Transactions on Algorithms*, 12:Art. 31, 18, 2016. doi:10.1145/2885499.

22 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701–717, 1980.

23 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, 3rd edition, 2013.

24 Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms ESA 2015: 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 1037–1049. Springer Berlin Heidelberg, 2015. doi:10.1007/978-3-662-48350-3_86.

25 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. International Symposium on Symbolic and Algebraic Computati on*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226, Berlin, 1979. Springer.

Improved Hardness for Cut, Interdiction, and Firefighter Problems*†

Euiwoong Lee

Carnegie Mellon University, Pittsburgh, PA, USA
euiwoonl@cs.cmu.edu

Abstract

We study variants of the classic *s-t* cut problem and prove the following improved hardness results assuming the Unique Games Conjecture (UGC).

- For *Length-Bounded Cut* and *Shortest Path Interdiction*, we show that both problems are hard to approximate within any constant factor, even if we allow bicriteria approximation. If we want to cut vertices or the graph is directed, our hardness ratio for Length-Bounded Cut matches the best approximation ratio up to a constant. Previously, the best hardness ratio was 1.1377 for Length-Bounded Cut [4] and 2 for Shortest Path Interdiction [24].
- For any constant $k \geq 2$ and $\epsilon > 0$, we show that *Directed Multicut* with k source-sink pairs is hard to approximate within a factor $k - \epsilon$. This matches the trivial k-approximation algorithm. By a simple reduction, our result for $k = 2$ implies that *Directed Multiway Cut* with two terminals (also known as *s-t Bicut*) is hard to approximate within a factor $2 - \epsilon$, matching the trivial 2-approximation algorithm.
- Assuming a variant of the UGC (implied by another variant of Bansal and Khot [6]), we prove that it is hard to approximate *Resource Minimization Fire Containment* within any constant factor. Previously, the best hardness ratio was 2 [28]. For directed layered graphs with b layers, our hardness ratio $\Omega(\log b)$ matches the best approximation algorithm [3, 9].

Our results are based on a general method of *converting an integrality gap instance to a length-control dictatorship test* for variants of the *s-t* cut problem, which may be useful for other problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases length bounded cut, shortest path interdiction, multicut, firefighter, unique games conjecture

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.92

1 Introduction

One of the most important implications of the Unique Games Conjecture (UGC, [25]) is the results of Khot et al. [26] and Raghavendra [40], which say that for any maximum constraint satisfaction problem (Max-CSP), an integrality gap instance of the standard semidefinite programming (SDP) relaxation can be converted to the NP-hardness result with the same gap. These results initiated the study of beautiful connections between power of convex relaxations and hardness of approximation, from which surprising results for both subjects have been discovered.

* The full version of this paper is available as [30], https://arxiv.org/abs/1607.05133.
† Supported by the Samsung Scholarship, the Simons Award for Graduate Students in TCS, and Venkat Guruswami's NSF CCF-1115525.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 92; pp. 92:1–92:14


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

While their results hold for problems in Max-CSPs, the framework of *converting an integrality gap instance to hardness* has been successfully applied to covering and graph cut problems. For graph cut problems, Manokaran et al. [34] showed that for *Undirected Multiway Cut* and its generalizations, an integrality gap of the standard linear programming (LP) relaxation implies the hardness result assuming the UGC. Their result is further generalized by Ene et al. [16] by formulating them as *Min-CSPs*. In addition, Kumar et al. [29] studied *Strict CSPs* and showed the same phenomenon for the standard LP relaxation.

One of the limitations of the previous CSP-based transformations from LP gap instances to hard instances is based on the fact that they do not usually preserve the desired structure of the constraint hypergraph.[1] For example, consider the *Length-Bounded Edge Cut* problem where the input consists of a graph $G = (V, E)$, two vertices $s, t \in V$, and a constant $l \in \mathbb{N}$, and the goal is to remove the fewest edges to ensure there is no path from s to t of length less than l. This problem can be viewed as a special case of *Hypergraph Vertex Cover* (HVC) by viewing each edge as a vertex of a hypergraph and creating a hyperedge for every s-t path of length less than l. While HVC is in turn a Strict CSP, its integrality gap instance cannot be converted to hardness using Kumar et al. [29] as a black-box, since the set of hyperedges created in the resulting hard instance is not guaranteed to correspond to the set of short s-t paths of some graph.

For Undirected Multiway Cut, Manokaran et al. [34] bypassed this difficulty by using 2-ary constraints so that the resulting constraint hypergraph becomes a graph again. For *Undirected Node-weighted Multiway Cut*, Ene et al. [16] used the equivalence to *Hypergraph Multiway Cut* [38] so that the resulting hypergraph does not need to satisfy additional structure. These problems are then formulated as a Min-CSP by using many labels which are supposed to represent different connected components. However, these Min-CSP based techniques often require nontrivial problem-specific ideas and do not seem to be easily generalized to many other cut problems.

We study variants of the classical s-t cut problem in both directed and undirected graphs that have been actively studied. We prove the optimal hardness or the first super-constant hardness for them. See Section 1.1 for the definitions of the problems and our results. All our results are based on the general framework of converting an integrality gap instance to a *length-control dictatorship test*. The structure of our length-control dictatorship tests allows us to naturally convert an integrality gap instance for the basic LP for various cut problems to hardness based on the UGC. Section 1.2 provides more detailed intuition of this framework. We believe that our framework is general and will be useful to prove tight inapproximability of other cut problems.

1.1 Problems and Results

Length-Bounded Cut and Shortest Path Interdiction. The *Length-Bounded Cut* problem is a natural variant of s-t cut, where given a graph (directed or undirected), $s, t \in V$, and an integer l, we only want to cut s-t paths of length strictly less than l.[2] Its practical motivation is based on the fact that in most communication / transportation networks, short paths are preferred to be used to long paths [32].

Lovász et al. [31] gave an exact algorithm for Length-Bounded Vertex Cut ($l \leq 5$) in undirected graphs. Mahjoub and McCormick [32] proved that Length-Bounded Edge Cut

[1] One of notable exceptions we are aware is the result of Guruswami et al. [21], using Kumar et al. [29] to show that k-Uniform k-Partite Hypergraph Vertex Cover is hard to approximate within a factor $\frac{k}{2} - \epsilon$ for any $\epsilon > 0$.

[2] It is more conventional to cut s-t paths of length *at most* l. We use this slightly nonconventional way to be more consistent with Shortest Path Interdiction.

admits an exact polynomial time algorithm for $l \leq 4$ in undirected graphs. Baier et al. [4] showed that both Length-Bounded Vertex Cut ($l > 5$) and Length-Bounded Edge Cut ($l > 4$) are NP-hard to approximate within a factor 1.1377. They presented $O(\min(l, \frac{n}{l})) = O(\sqrt{n})$-approximation algorithm for Length-Bounded Vertex Cut and $O(\min(l, \frac{n^2}{l^2}, \sqrt{m})) = O(n^{2/3})$-approximation algorithm for Length-Bounded Edge Cut, with matching LP gaps. Length-Bounded Cut problems have been also actively studied in terms of their fixed parameter tractability [19, 15, 8, 17].

If we exchange the roles of the objective k and the length bound l, the problem becomes *Shortest Path Interdiction*, where we want to maximize the length of the shortest s-t path after removing at most k vertices or edges. It is also one of the central problems in a broader class of *interdiction* problems, where an *attacker* tries to remove some edges or vertices to destroy a desirable property (e.g., short s-t distance, large s-t flow, cheap MST) of a network (see the survey of [42]). The study of Shortest Path Interdiction started in 1980's when the problem was called as the *k-most-vital-arcs* problem [14, 33, 5] and proved to be NP-hard [5]. Khachiyan et al. [24] proved that it is NP-hard to approximate within a factor less than 2. While many heuristic algorithms were proposed [23, 7, 35] and hardness in planar graphs [39] was shown, whether the general version admits a constant factor approximation was still unknown.

Given a graph $G = (V, E)$ and $s, t \in V$, let $\mathsf{dist}(G)$ be the length of the shortest s-t path. For $V' \subseteq V$, let $G \setminus V'$ be the subgraph induced by $V \setminus V'$. For $E' \subseteq E$, we use the same notation $G \setminus E'$ to denote the subgraph $(V, E \setminus E')$. We primarily study undirected graphs. We first present our results for the vertex version of both problems (collectively called as *Short Path Vertex Cut* onwards).

▶ **Theorem 1.** *Assuming the Unique Games Conjecture, for infinitely many values of constant $l \in \mathbb{N}$, the following three tasks are NP-hard: Given an undirected graph $G = (V, E)$ and $s, t \in V$ where there exists $C^* \subseteq V \setminus \{s, t\}$ such that $\mathsf{dist}(G \setminus C^*) \geq l$,*

1. *Find $C \subseteq V \setminus \{s, t\}$ such that $|C| \leq \Omega(l) \cdot |C^*|$ and $\mathsf{dist}(G \setminus C) \geq l$.*
2. *Find $C \subseteq V \setminus \{s, t\}$ such that $|C| \leq |C^*|$ and $\mathsf{dist}(G \setminus C) \geq O(\sqrt{l})$.*
3. *Find $C \subseteq V \setminus \{s, t\}$ such that $|C| \leq \Omega(l^{\frac{\epsilon}{2}}) \cdot |C^*|$ and $\mathsf{dist}(G \setminus C) \geq O(l^{\frac{1+\epsilon}{2}})$ for some $0 < \epsilon < 1$.*

The first result shows that Length Bounded Vertex Cut is hard to approximate within a factor $\Omega(l)$. This matches the best $O(l)$-approximation [4] when l is a constant. The second result shows that Shortest Path Vertex Interdiction is hard to approximate with in a factor $\Omega(\sqrt{\mathsf{OPT}})$, and the third result rules out *bicriteria approximation* – for any constant c, it is hard to approximate both l and $|C^*|$ within a factor of c.

The above results hold for directed graphs by definition. Our hard instances will have a natural *layered* structure, so it can be easily checked that the same results (up to a constant) hold for directed acyclic graphs. Since one vertex can be split as one directed edge, the same results hold for the edge version in directed acyclic graphs.

For Length-Bounded Edge Cut and Shortest Path Edge Interdiction in undirected graphs (collectively called *Short Path Edge Cut* onwards), we prove the following theorems.

▶ **Theorem 2.** *Assuming the Unique Games Conjecture, for infinitely many values of constant $l \in \mathbb{N}$, the following three tasks are NP-hard: Given an undirected graph $G = (V, E)$ and $s, t \in V$ where there exists $C^* \subseteq E$ such that $\mathsf{dist}(V \setminus C^*) \geq l$,*

1. *Find $C \subseteq E$ such that $|C| \leq \Omega(\sqrt{l}) \cdot |C^*|$ and $\mathsf{dist}(G \setminus C) \geq l$.*
2. *Find $C \subseteq E$ such that $|C| \leq |C^*|$ and $\mathsf{dist}(G \setminus C) \geq l^{\frac{2}{3}}$.*
3. *Find $C \subseteq E$ such that $|C| \leq \Omega(l^{\frac{2\epsilon}{3}}) \cdot |C^*|$ and $\mathsf{dist}(G \setminus C) \geq O(l^{\frac{2+2\epsilon}{3}})$ for some $0 < \epsilon < \frac{1}{2}$.*

Our hardness factors for the undirected edge versions, $\Omega(\sqrt{l})$ for Length-Bounded Edge Cut and $\Omega(\sqrt[3]{\mathsf{OPT}})$ for Shortest Path Edge Interdiction, are slightly weaker than those for their vertex counterparts, but we are not aware of any approximation algorithm specialized for the undirected edge versions. It is an interesting open problem whether there exist better approximation algorithms for the undirected edge versions.

Directed Multicut and Directed Multiway Cut. Given a directed graph and two vertices s and t, one of the most natural variants of s-t cut is to remove the fewest edges to ensure that there is no directed path from s to t and no directed path from t to s. This problem is known as *s-t Bicut* and admits the trivial 2-approximation algorithm by computing the minimum s-t cut and t-s cut.

Directed Multiway Cut is a generalization of s-t Bicut that has been actively studied. Given a directed graph with k terminals $s_1, \ldots, s_k$, the goal is to remove the fewest number of edges such that there is no path from s_i to s_j for any $i \neq j$. Directed Multiway Cut also admits 2-approximation [37, 11]. If k is allowed to increase polynomially with n, there is a simple reduction from Vertex Cover that shows $(2-\epsilon)$-approximation is hard under the UGC [18, 27].

Directed Multiway Cut can be further generalized to *Directed Multicut*. Given a directed graph with k source-sink pairs $(s_1, t_1), \ldots, (s_k, t_k)$, the goal is to remove the fewest number of edges such that there is no path from s_i to t_i for any i. Computing the minimum s_i-t_i cut for all i separately gives the trivial k-approximation algorithm. Chuzhoy and Khanna [13] showed Directed Multicut is hard to approximate within a factor $2^{\Omega(\log^{1-\epsilon} n)} = 2^{\Omega(\log^{1-\epsilon} k)}$ when k is polynomially growing with n. Agarwal et al. [2] showed $\tilde{O}(n^{\frac{11}{23}})$-approximation algorithm, which improves the trivial k-approximation when k is large.

Chekuri and Madan [11] showed simple approximation-preserving reductions from Directed Multicut with $k = 2$ to s-t Bicut (the other direction is trivially true), and *(Undirected) Node-weighted Multiway Cut* with $k = 4$ to s-t Bicut. Since Node-weighted Multiway Cut with $k = 4$ is hard to approximate within a factor $1.5 - \epsilon$ under the UGC [16] (matching the algorithm of Garg et al. [18]), the same hardness holds for s-t Bicut, Directed Multiway Cut, and Directed Multicut for constant k. To the best of our knowledge, $1.5 - \epsilon$ is the best hardness factor for constant k even assuming the UGC. In the same paper, Chekuri and Madan [11] asked whether a factor $2 - \epsilon$ hardness holds for s-t Bicut under the UGC.

We prove that for any constant $k \geq 2$, the trivial k-approximation for Directed Multicut might be optimal. Our result for $k = 2$ gives the optimal hardness result for s-t Bicut, answering the question of Chekuri and Madan.

▶ **Theorem 3.** *Assuming the Unique Games Conjecture, for every $k \geq 2$ and $\epsilon > 0$, Directed Multicut with k source-sink pairs is NP-hard to approximate within a factor $k - \epsilon$.*

▶ **Corollary 4.** *Assuming the Unique Games Conjecture, for any $\epsilon > 0$, s-t Bicut is hard to approximate within a factor $2 - \epsilon$.*

▶ Remark. Chekuri and Madan [12] obtained an independent and different proof of Theorem 3.

RMFC. *Resource Minimization for Fire Containment (RMFC)* is a problem closely related to Length-Bounded Cut with the additional notion of time. Given a graph G, a vertex s, and a subset T of vertices, consider the situation where fire starts at s on Day 0. For each Day i ($i \geq 1$), we can *save* at most k vertices, and the fire spreads from currently burning vertices to its unsaved neighbors. Once a vertex is burning or saved, it remains so from then onwards. The process is terminated when the fire cannot spread anymore. RMFC asks to

find a strategy to save k vertices each day with the minimum k so that no vertex in T is burnt. These problems model the spread of epidemics or ideas through a social network, and have been actively studied recently [9, 3, 1, 10].

RMFC, along with other variants, is first introduced by Hartnell [22]. Another well-studied variant is called the *Firefighter* problem, where we are only given $s \in V$ and want to maximize the number of vertices that are not burnt at the end. It is known to be NP-hard to approximate within a factor $n^{1-\epsilon}$ for any $\epsilon > 0$ [3]. King and MacGillivray [28] proved that RMFC is hard to approximate within a factor less than 2. Anshelevich et al. [3] presented an $O(\sqrt{n})$-approximation algorithm for general graphs, and Chalermsook and Chuzhoy [9] showed that RMFC admits $O(\log^* n)$-approximation in trees. Very recently, the approximation ratio in trees has been improved to $O(1)$ [1]. Both Anshelevich et al. [3] and Chalermsook and Chuzhoy [9] independently studied directed layer graphs with b layers, showing $O(\log b)$-approximation.

Our final result on RMFC assumes a variant of the Unique Games Conjecture which is not known to be equivalent to the original UGC. Given a bipartite graph as an instance of Unique Games, it states that in the completeness case, all constraints incident on $(1-\epsilon)$ fraction of vertices in one side are satisfied, and in the soundness case, in addition to having a low value, every $\frac{1}{10}$ fraction of vertices on one side have at least a $\frac{9}{10}$ fraction of vertices on the other side as neighbors. Our conjecture is implied by the conjecture of Bansal and Khot [6] that is used to prove the hardness of *Minimizing Weighted Completion Time with Precedence Constraints* and requires a more strict expansion condition. See [30] for the exact statement.

▶ **Theorem 5.** *Assuming Conjecture 7.5 of [30], it is NP-hard to approximate RMFC in undirected graphs within any constant factor.*

Again, our reduction has a natural layered structure and the result holds for directed layered graphs. With b layers, we prove that it is hard to approximate with in a factor $\Omega(\log b)$, matching the best approximation algorithms [9, 3].

1.2 Techniques

All our results are based on a general method of *converting an integrality gap instance to a dictatorship test*. This method has been successfully applied by Raghavendra [40] for Max-CSPs, Manokaran et al. [34] and Ene et al. [16] for Multiway Cut and Min CSPs, and Kumar et al. [29] for strict CSPs, and by Guruswami et al. [21] for k-uniform k-partite Hypergraph Vertex Cover, and Chekuri and Madan [12] for Directed Multicut. As mentioned in the introduction, the previous CSP-based results do not generally preserve the structure of constraint hypergraphs or use ingenious and specialized tricks to reduce the problem to a CSP.

We bypass this difficulty by constructing a special class of dictatorship tests that we call *length-control dictatorship tests*. Consider a meta-problem where given a directed graph $G = (V, E)$, some terminal vertices, and a set $\mathcal{P}$ of *desired paths* between terminals, we want to remove the fewest number of non-terminal vertices to cut every path in $\mathcal{P}$. The integrality gap instances we use in this work [41, 4, 32, 9] share the common feature that every $p \in \mathcal{P}$ is of length at least r, and the fractional solution cuts $\frac{1}{r}$ fraction of each non-terminal vertex so that each path $p \in \mathcal{P}$ is cut. This gives a good LP value, and additional arguments are required to ensure that there is no efficient integral cut.

Given such an integrality gap instance, we construct our dictatorship test instance as follows. We replace every non-terminal vertex by a *hypercube* $\mathbb{Z}_r^R$ and put edges such that

for two vertices (v, x) and (w, y) where $v, w \in V$ and $x, y \in \mathbb{Z}_r^R$, there is an edge from (v, x) to (w, y) if (1) $(v, w) \in E$ and (2) $y_j = x_j + 1$ for all $j \in [R]$. The set of desired paths $\mathcal{P}'$ is defined to be $\{(s, (v_1, x_1), \ldots, (v_l, x_l), t) : (s, v_1, \ldots, v_l, t) \in \mathcal{P}\}$ (s, t denote some terminals). Note that each path in $\mathcal{P}'$ is also of length at least r. We want to ensure that in the *completeness case* (i.e., every hypercube reveals the same *influential coordinate*), there is a very efficient cut, while in the *soundness case* (i.e., no hypercube reveals an influential coordinate), there is no such efficient cut.

In the completeness case, let $q \in [R]$ be an influential coordinate. For each vertex (v, x) where $v \in V, x \in \mathbb{Z}_r^R$, remove (v, x) if $x_q = 0$. Consider a desired path $p = (s, (v_1, x_1), \ldots, (v_l, x_l), t) \in \mathcal{P}'$ for some terminals s, t and some $v_j \in V, x_j \in \mathbb{Z}_r^R$ $(1 \le j \le l)$, and let $y_j = (x_j)_q$. By our construction, $y_{j+1} = y_j + 1$ for $0 \le j < l$. Since p is desirable, $l \ge r$, so there exists j such that $y_j = (x_j)_q = 0$, but (v_j, x_j) is already removed by our previous definition. Therefore, every desired path is cut by this vertex cut. Note that this cut is integral and cuts exactly $\frac{1}{r}$ fraction of non-terminal vertices. This corresponds to the fractional solution to the gap instance that cuts $\frac{1}{r}$ fraction of every vertex.

For the soundness analysis, our final dictatorship test has additional *noise* vertices and edges to the test defined above. If no hypercube reveals an influential coordinate, the standard application of the invariance principle [36] proves that we can always take an edge between two hypercubes unless we almost completely cut one hypercube. We can then invoke the proof for the integrality gap instance to show that there is no efficient cut.

This idea is implicitly introduced by the work of Svensson [43] for Feedback Vertex Set (FVS) and DAG Vertex Deletion (DVD) by applying the *It ain't over till it's over theorem* to ingeniously constructed dictatorship tests with auxiliary vertices. Guruswami and Lee [20] gave a simpler construction and a new proof using the invariance principle instead of the It ain't over till it's over theorem. Our results are based on the observation that length-control dictatorship tests and LP gap instances *fool* algorithms in a similar way for various cut problems as mentioned above, so that the previous LP gap instances can be plugged into our framework to prove matching hardness results.

This method for the above meta-problem can be almost directly applied to Directed Multicut. For Length-Bounded Cut and RMFC in undirected graphs, we use the fact that the known integrality gap instances have a natural layered structure with s in the first layer and t in the last layer. Every edge is given a natural orientation, and the similar analysis can be applied. For Length-Bounded Cut, another set of edges called *long edges* are added to the dictatorship test. More technical work is required for edge cut versions in undirected graphs (Short Path Edge Cut), and the notion of time (RMFC).

Our framework seems general enough so that they can be applied to integrality gap instances to give strong hardness results. It would be interesting to further abstract this method of converting integrality gap instances to length-bounded dictatorship tests, as well as to apply it to other problems whose approximability is not well-understood.

2 Preliminaries

Graph Terminologies. Depending on whether we cut vertices or edges, we introduce weight $\mathsf{wt}(v)$ for each vertex v, or weight $\mathsf{wt}(e)$ for each edge e. Some weights can be ∞, which means that some vertices or edges cannot be cut. For vertex-weighted graphs, we naturally have $\mathsf{wt}(s) = \mathsf{wt}(t) = \infty$. To reduce the vertex-weighted version to the unweighted version, we duplicate each vertex according to its weight and replace each edge by a complete bipartite graph between corresponding copies. To reduce the edge-weighted version to the unweighted

version, we replace a single edge with parallel edges according to its weight. To reduce to simple graphs, we split each parallel into two edges by introducing a new vertex.

For the Length-Bounded Cut problems, we also introduce length $\mathsf{len}(e)$ for each edge e. It can be also dealt with serially splitting an edge according to its weight. We allow weights to be rational numbers, but as our hardness results are stated in terms of the length, all lengths in this work will be a positive integer.

For a path p, depending on the context, we abuse notation and interpret it as a set of edges or a set of vertices. The length of p is always defined to be the number of edges.

Gaussian Bounds for Correlated Spaces. We introduce the standard tools on correlated spaces from Mossel [36]. Given a probability space (Ω, μ) (we always consider finite probability spaces), let $\mathcal{L}(\Omega)$ be the set of functions $\{f : \Omega \to \mathbb{R}\}$ and for an interval $I \subseteq \mathbb{R}$, $\mathcal{L}_I(\Omega)$ be the set of functions $\{f : \Omega \to I\}$. For a subset $S \subseteq \Omega$, define *measure* of S to be $\mu(S) := \sum_{\omega \in S} \mu(\omega)$. A collection of probability spaces are said to be correlated if there is a joint probability distribution on them. We will denote k correlated spaces $\Omega_1, \dots, \Omega_k$ with a joint distribution μ as $(\Omega_1 \times \cdots \times \Omega_k, \mu)$.

Given two correlated spaces $(\Omega_1 \times \Omega_2, \mu)$, we define the correlation between Ω_1 and Ω_2 by

$$\rho(\Omega_1, \Omega_2; \mu) := \sup \{\mathsf{Cov}[f, g] : f \in \mathcal{L}(\Omega_1), g \in \mathcal{L}(\Omega_2), \mathsf{Var}[f] = \mathsf{Var}[g] = 1\}.$$

Given a probability space (Ω, μ) and a function $f \in \mathcal{L}(\Omega)$ and $p \in \mathbb{R}^+$, let $\|f\|_p := \mathbb{E}_{x \sim \mu}[|f(x)|^p]^{1/p}$.

Consider a product space $(\Omega^R, \mu^{\otimes R})$ and $f \in \mathcal{L}(\Omega^R)$. The *Efron-Stein decomposition* of f is given by

$$f(x_1, \dots, x_R) = \sum_{S \subseteq [R]} f_S(x_S)$$

where (1) f_S depends only on x_S and (2) for all $S \not\subseteq S'$ and all $x_{S'}$, $\mathbb{E}_{x' \sim \mu^{\otimes R}}[f_S(x') | x'_{S'} = x_{S'}] = 0$. The *influence* of the ith coordinate on f is defined by

$$\mathsf{Inf}_i[f] := \mathop{\mathbb{E}}_{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_R} [\mathop{\mathsf{Var}}_{x_i}[f(x_1, \dots, x_R)].$$

The influence has a convenient expression in terms of the Efron-Stein decomposition.

$$\mathsf{Inf}_i[f] = \| \sum_{S : i \in S} f_S \|_2^2 = \sum_{S : i \in S} \|f_S\|_2^2.$$

We also define the *low-degree influence* of the ith coordinate.

$$\mathsf{Inf}_i^{\le d}[f] := \sum_{S : i \in S, |S| \le d} \|f_S\|_2^2.$$

For $a, b \in [0, 1]$ and $\rho \in (0, 1)$, let

$$\Gamma_\rho(a, b) := \Pr[X \le \Phi^{-1}(a), Y \ge \Phi^{-1}(1 - b)],$$

where X and Y are ρ-correlated standard Gaussian variables and Φ denotes the cumulative distribution function of a standard Gaussian. The following theorem bounds the product of two functions that do not share an influential coordinate in terms of their Gaussian counterparts.

▶ **Theorem 6** (Theorem 6.3 and Lemma 6.6 of [36]). *Let $(\Omega_1 \times \Omega_2, \mu)$ be correlated spaces such that the minimum nonzero probability of any atom in $\Omega_1 \times \Omega_2$ is at least α and such that $\rho(\Omega_1, \Omega_2; \mu) \le \rho$. Then for every $\epsilon > 0$ there exist τ, d depending on ϵ and α such that if $f : \Omega_1^R \to [0, 1], g : \Omega_2^R \to [0, 1]$ satisfy $\min(\mathsf{Inf}_i^{\le d}[f], \mathsf{Inf}_i^{\le d}[g]) \le \tau$ for all i, then $\mathbb{E}_{(x,y) \in \mu^{\otimes R}}[f(x)g(y)] \ge \Gamma_\rho(\mathbb{E}_x[f], \mathbb{E}_y[g]) - \epsilon$.*

Organization. The dictatorship tests for Short Path Edge Cut and Short Path Vertex Cut are presented in Section 3 and 4 respectively. Dictatorship tests for RMFC and Directed Multicut, as well as the reduction from Unique Games based on these tests, will appear in the full version of this paper [30].

3 Short Path Edge Cut

We propose our dictatorship test for Short Path Edge Cut that will be used for proving Unique Games hardness. It is parameterized by positive integers a, b, r, R. It is inspired by the integrality gap instances by Baier et al. [4] Mahjoub and and McCormick [32], and made such that the edge cuts that correspond to *dictators* behave the same as the fractional solution that cuts $\frac{1}{r}$ fraction of every edge. All graphs in this section are undirected.

For positive integers a, b, r, R, we construct $\mathcal{D}^{\mathsf{E}}_{a,b,r,R} = (V, E)$. Let $\Omega = \{0, \dots, r-1\}$, and $\mu : \Omega \mapsto [0,1]$ with $\mu(x) = \frac{1}{r}$ for each $x \in \Omega$. We also define a correlated probability space $(\Omega_1 \times \Omega_2, \nu)$ where both Ω_1, Ω_2 are copies of Ω. It is defined by the following process to sample $(x, y) \in \Omega^2$.

- Sample $x \in \{0, \dots, r-1\}$. Let $y = (x+1) \mod r$.
- With probability $1 - \frac{1}{r}$, output (x, y). Otherwise, resample $x, y \in \Omega$ independently and output (x, y).

Note that the marginal distribution of both x and y is equal to μ. Given $x = (x_1, \dots, x_R) \in \Omega^R$ and $y = (y_1, \dots, y_R) \in \Omega^R$, let $\nu^{\otimes R}(x, y) = \prod_{i=1}^{R} \nu(x_i, y_i)$. We define $\mathcal{D}^{\mathsf{E}}_{a,b,r,R} = (V, E)$ as follows.

- $V = \{s, t\} \cup \{v^i_x\}_{0 \le i \le b, x \in \Omega^R}$. Let v^i denote the set of vertices $\{v^i_x\}_{x \in \Omega^R}$.
- For any $x \in \Omega^R$, there is an edge from s to v^0_x and an edge from v^b_x to t, both with weight ∞ and length 1.
- For $0 \le i < b, x \in \Omega^R$, there is an edge (v^i_x, v^{i+1}_x) of length a and weight ∞. Call it a *long edge*.
- For any $0 \le i < b$ $x, y \in \Omega^R$, there is an edge (v^i_x, v^{i+1}_y) of length 1 and weight $\nu^{\otimes R}(x, y)$. Note that $\nu^{\otimes R}(x, y) > 0$ for any $x, y \in \Omega^R$. Call it a *short edge*. The sum of finite weights is b.

Completeness. We first prove that edge cuts that correspond to *dictators* behave the same as the fractional solution that gives $\frac{1}{r}$ to every short edge. Fix $q \in [R]$ and let E_q be the set of short edges defined by

$$E_q := \{(v^i_x, v^{i+1}_y) : 0 \le i < b, y_q \ne x_q + 1 \mod R \text{ or } (x_q, y_q) = (0, 1)\}.$$

When $(x, y) \in \Omega_1 \times \Omega_2$ is sampled according to ν, the probability that $y_q \ne x_q + 1 \mod R$ or $(x_q, y_q) = (0, 1)$ is at most $\frac{2}{r}$. The total weight of E_q is $\frac{2b}{r}$.

▶ **Lemma 7.** *After removing edges in E_q, the length of the shortest path is at least $a(b-r+1)$.*

Proof. Let $p = (s, v^{i_1}_{x^1}, \dots, v^{i_z}_{x^z}, t)$ be a path from s to t where $i_j \in \{0, \dots, b\}$ and $x^j \in \Omega^R$ for each $1 \le j \le z$. Let $y_j := (x^j)_q \in \{0, \dots, r-1\}$ for each $1 \le j \le z$.

For each $1 \le j < z$, the edge (p_j, p_{j+1}) is either a long edge or a short edge, and either taken forward (i.e., $i_j < i_{j+1}$) or backward (i.e., $i_j > i_{j+1}$). Let $z_{\mathsf{LF}}, z_{\mathsf{SF}}, z_{\mathsf{LB}}, z_{\mathsf{SB}}$ be the number of long edges taken forward, short edges taken forward, long edges taken backward, and shot edges taken backward, respectively ($z_{\mathsf{LF}} + z_{\mathsf{SF}} + z_{\mathsf{LB}} + z_{\mathsf{SB}} = z - 1$). By considering how i_j changes,

$$z_{\mathsf{LF}} + z_{\mathsf{SF}} - z_{\mathsf{LB}} - z_{\mathsf{SB}} = b. \tag{1}$$

Consider how y_j changes. Taking a long edge does not change y_j. Taking a short edge forward increases y_j by 1 mod r, taking a short edge backward decreases y_j by 1 mod r. Since E_q is cut, y_j can never change from 0 to 1. This implies

$$z_{\mathsf{SF}} - z_{\mathsf{SB}} \leq r - 1. \tag{2}$$

(1) − (2) yields $z_{\mathsf{LF}} - z_{\mathsf{LB}} \geq b - r + 1$. The total length of p is at least $a \cdot z_{\mathsf{LF}} \geq a(b - r + 1)$. ◀

Soundness. We first bound the correlation $\rho(\Omega_1, \Omega_2; \nu)$. The following lemma of Wenner [44] gives a convenient way to bound the correlation.

▶ **Lemma 8** (Corollary 2.18 of [44]). *Let $(\Omega_1 \times \Omega_2, \delta\mu + (1-\delta)\mu')$ be two correlated spaces such that the marginal distribution of at least one of Ω_1 and Ω_2 is identical on μ and μ'. Then,*

$$\rho(\Omega_1, \Omega_2; \delta\mu + (1-\delta)\mu') \leq \sqrt{\delta \cdot \rho(\Omega_1, \Omega_2; \mu)^2 + (1-\delta) \cdot \rho(\Omega_1, \Omega_2; \mu')^2}.$$

When (x, y) is sampled from ν, they are completely independent with probability $\frac{1}{r}$. Therefore, we have $\rho := \rho(\Omega_1, \Omega_2; \nu) \leq \sqrt{1 - \frac{1}{r}}$. By Sheppard's Formula,

$$\Gamma_\rho(\frac{1}{2}, \frac{1}{2}) = \frac{1}{4} + \frac{1}{2\pi}\arcsin(-\rho) \geq \frac{1}{4} - \frac{1}{2\pi}\arccos(\frac{1}{\sqrt{r}}) = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} (\frac{1}{\sqrt{r}})^{2n+1} \geq \frac{1}{\sqrt{r}}.$$

Apply Theorem 6 ($\rho \leftarrow \rho, \alpha \leftarrow \frac{1}{r^3}, \epsilon \leftarrow \frac{\Gamma_\rho(\frac{1}{2},\frac{1}{2})}{3}$) to get τ and d. We will later apply this theorem with the parameters obtained here.

Fix an arbitrary subset $C \subseteq E$ of short edges. For $0 \leq i < b$, let $C_i = C \cap (v^i \times v^{i+1})$. Call a pair $(i, i+1)$ as the ith layer, and say it is blocked when $\nu^{\otimes R}(C_i) \geq \frac{\Gamma_\rho(\frac{1}{2},\frac{1}{2})}{2}$. Let b' be the number of blocked layers. For $0 \leq i \leq b$, let $S_i \subseteq v^i$ be such that $x \in S_i$ if there exists a path $(s, p_0, \ldots, p_i = v^i_x)$ such that

- For $0 \leq i' \leq i$, $p_{i'} \in v^{i'}$.
- For $0 \leq i' < i$, $(p_{i'}, p_{i'+1})$ is short if and only if the i'th layer is unblocked.

Let $f_i : \Omega^R \mapsto [0, 1]$ be the indicator function of S_i. We prove that if none of f_i reveals any *influential coordinate*, S_b is nonempty, implying that there exists a path using b' long edges and $b - b'$ short edges. Therefore, even after removing edges in C, the length of the shortest path is at most $2 + ab' + (b - b')$.

▶ **Lemma 9.** *Suppose that for any $0 \leq i \leq b$ and $1 \leq j \leq R$, $\mathsf{Inf}_j^{\leq d}[f_i] \leq \tau$. Then $S_b \neq \emptyset$.*

Proof. Assume towards contradiction that $S_b = \emptyset$. Since $S_0 = \Omega^R$ and $S_i = S_{i+1}$ if the ith layer is blocked (and we use long edges), there must exist i such that the ith layer is unblocked and $\mu^{\otimes R}(S_i) \geq \frac{1}{2}, \mu^{\otimes R}(S_{i+1}) < \frac{1}{2}$. All short edges between S_i and $v^{i+1} \setminus S_{i+1}$ are in C_i. Theorem 6 implies that $\nu^{\otimes R}(C_i) > \frac{2}{3}\Gamma_\rho(\frac{1}{2}, \frac{1}{2})$. This contradicts the fact that the ith layer is unblocked. ◀

In summary, in the completeness case, if we cut edges of total weight $k := k(a, b, r) = \frac{2b}{r}$, the length of the shortest path is at least $l := l(a, b, r) = a(b - r + 1)$. In the soundness case, even after cutting edges of total weight k', at most $\frac{2k'}{\Gamma_\rho(\frac{1}{2},\frac{1}{2})} \leq 2k'\sqrt{r}$ layers are blocked, the length of the shortest path is at most $l' = 2 + (b - 2k'\sqrt{r}) + 2ak'\sqrt{r}$.

- Let $a = 4, b = 2r - 1$ so that $k \leq 4, l = 4r$. Requiring $l' \geq l$ results in $k' = \Omega(\sqrt{r})$, giving a gap of $\Omega(\sqrt{r}) = \Omega(\sqrt{l})$ between the completeness case and the soundness case for Length-Bounded Edge Cut.
- Let $a = \sqrt{r}, b = 2r - 1$ so that $k \leq 4, l = r^{1.5}$. Requiring $k' \leq 4$ results in $l' = O(r)$, giving a gap of $\Omega(\sqrt{r}) = \Omega(l^{1/3})$ for Shortest Path Interdiction. Generally, $k' \leq O(r^\epsilon)$ results in $l' \leq O(r^{1+\epsilon})$, giving an $(O(r^\epsilon), O(r^{1/2-\epsilon}))$-bicriteria gap for any $\epsilon \in (0, \frac{1}{2})$.

4 Short Path Vertex Cut

We propose our dictatorship test for Short Path Vertex Cut that will be used for proving Unique Games hardness. It is parameterized by positive integers a, b, r, R and small $\epsilon > 0$. It is inspired by the integrality gap instances by Baier et al. [4] Mahjoub and and McCormick [32], and made such that the vertex cuts that correspond to *dictators* behave the same as the fractional solution that cuts $\frac{1}{r}$ fraction of every vertex. All graphs in this section are undirected.

For positive integers a, b, r, R, and $\epsilon > 0$, define $\mathcal{D}^{\mathsf{V}}_{a,b,r,R,\epsilon} = (V, E)$ be the graph defined as follows. Consider the probability space (Ω, μ) where $\Omega := \{0, \dots, r-1, *\}$, and $\mu : \Omega \mapsto [0, 1]$ with $\mu(*) = \epsilon$ and $\mu(x) = \frac{1-\epsilon}{r}$ for $x \neq *$.

- $V = \{s, t\} \cup \{v^i_x\}_{0 \leq i \leq b, x \in \Omega^R}$. Let v^i denote the set of vertices $\{v^i_x\}_x$.
- For $0 \leq i \leq b$ and $x \in \Omega^R$, $\mathsf{wt}(v^i_x) = \mu^{\otimes R}(x)$. Note that the sum of weights is $b + 1$.
- For any $0 \leq i \leq b$, there are edges from s to each vertex in v_i with length $ai + 1$ and edges from each vertex in v_i to t with length $(b - i)a + 1$.
- For $x, y \in \Omega^R$, we call that x and y are *compatible* if
  - For any $1 \leq j \leq R$: $[y_j = (x_j + 1) \mod r]$ or $[y_j = *]$ or $[x_j = *]$.
- For any $0 \leq i < b$ and compatible $x, y \in \Omega^R$, we have an edge (v^i_x, v^{i+1}_y) of length 1 (called a *short edge*).
- For any i, j such that $0 \leq i < j - 1 < b$ and compatible $x, y \in \Omega^R$, we have an edge (v^i_x, v^j_y) of length $(j - i)a$ (called a *long edge*).

Completeness. We first prove that vertex cuts that correspond to *dictators* behave the same as the fractional solution that gives $\frac{1}{r}$ to every vertex. For any $q \in [R]$, let $V_q := \{v^i_x : 0 \leq i \leq b, x_q = * \text{ or } 0\}$. Note that the total weight of V_q is $(b + 1)(\epsilon + \frac{1-\epsilon}{r})$.

▶ **Lemma 10.** *After removing vertices in V_q, the length of the shortest path is at least $a(b - r + 2)$.*

Proof. Let $p = (s, v^{i_1}_{x^1}, \dots, v^{i_z}_{x^z}, t)$ be a path from s to t where $i_j \in \{0, \dots, b\}$ and $x^j \in \Omega^R$ for each $1 \leq j \leq z$. Let $y_j := (x^j)_q \in \{0, \dots, r-1\}$ for each $1 \leq j \leq z$.

For each $1 \leq j < z$, the edge $(v^{i_j}_{x^j}, v^{i_{j+1}}_{x^{j+1}})$ is either a long edge or a short edge, and either taken forward (i.e., $i_j < i_{j+1}$) or backward (i.e., $i_j > i_{j+1}$). Let $z_{\mathsf{LF}}, z_{\mathsf{SF}}, z_{\mathsf{LB}}, z_{\mathsf{SB}}$ be the number of long edges taken forward, short edges taken forward, long edges taken backward, and shot edges taken backward, respectively ($z_{\mathsf{LF}} + z_{\mathsf{SF}} + z_{\mathsf{LB}} + z_{\mathsf{SB}} = z - 1$). For $1 \leq j \leq z_{\mathsf{LF}}$ (resp. z_{LB}), consider the jth long edge taken forward (resp. backward) – it is $(v^{i_{j'}}_{x^{j'}}, v^{i_{j'+1}}_{x^{j'+1}})$ for some j'. Let s^{F}_j (resp. s^{B}_j) be $|i_{j'} - i_{j'+1}|$. The following equality holds by observing how i_j changes.

$$i_1 + \sum_{j=1}^{z_{\mathsf{LF}}} s^{\mathsf{F}}_j + z_{\mathsf{SF}} - \sum_{j=1}^{z_{\mathsf{LB}}} s^{\mathsf{B}}_j - z_{\mathsf{SB}} = i_z \quad \Rightarrow \quad i_1 + \sum_{j=1}^{z_{\mathsf{LF}}} s^{\mathsf{F}}_j + z_{\mathsf{SF}} - z_{\mathsf{LB}} - z_{\mathsf{SB}} - i_z \geq 0. \tag{3}$$

Consider how y_j changes. Taking any edge forward increases y_j, and taking any edge backward decreases y_j. Since y_j can never be 0 or $*$, we can conclude that

$$z_{\mathsf{LF}} + z_{\mathsf{SF}} - z_{\mathsf{LB}} - z_{\mathsf{SB}} \leq r - 2. \tag{4}$$

(3) − (4) yields

$$i_1 - i_z + \sum_{j=1}^{z_{\mathsf{LF}}} (s_j^{\mathsf{F}} - 1) \geq 2 - r \quad \Rightarrow \quad i_1 - i_z + \sum_{j=1}^{z_{\mathsf{LF}}} s_j^{\mathsf{F}} \geq 2 - r. \tag{5}$$

The total length of p is

$$\begin{aligned} & 2 + a\big(i_1 + b - i_z + \sum_{j=1}^{z_{\mathsf{LF}}} s_j^{\mathsf{F}} + \sum_{j=1}^{z_{\mathsf{LB}}} s_j^{\mathsf{B}}\big) + z_{\mathsf{SF}} + z_{\mathsf{SB}} \\ \geq \; & a\big(i_1 + b - i_z + \sum_{j=1}^{z_{\mathsf{LF}}} s_j^{\mathsf{F}}\big) \\ \geq \; & a(b - r + 2). \end{aligned}$$

◀

Soundness. To analyze soundness, we define a correlated probability space $(\Omega_1 \times \Omega_2, \nu)$ where both Ω_1, Ω_2 are copies of $\Omega = \{0, \ldots, r-1, *\}$. It is defined by the following process to sample $(x, y) \in \Omega^2$.

- Sample $x \in \{0, \ldots, r-1\}$. Let $y = (x+1) \mod r$.
- Change x to $*$ with probability ϵ. Do the same for y independently.

Note that the marginal distribution of both x and y is equal to μ. Assuming $\epsilon < \frac{1}{2r}$, the minimum probability of any atom in $\Omega_1 \times \Omega_2$ is ϵ^2. Furthermore, in our correlated space, $\nu(x, *) > 0$ for all $x \in \Omega_1$ and $\nu(*, x) > 0$ for all $x \in \Omega_2$. We use the following lemma to bound the correlation.

▶ **Lemma 11** (Lemma 2.9 of [36]). *Let $(\Omega_1 \times \Omega_2, \mu)$ be two correlated spaces such that the probability of the smallest atom in $\Omega_1 \times \Omega_2$ is at least $\alpha > 0$. Define a bipartite graph $G = (\Omega_1 \cup \Omega_2, E)$ where $(a, b) \in \Omega_1 \times \Omega_2$ satisfies $(a, b) \in E$ if $\mu(a, b) > 0$. If G is connected, then $\rho(\Omega_1, \Omega_2; \mu) \leq 1 - \frac{\alpha^2}{2}$.*

Therefore, we can conclude that $\rho(\Omega_1, \Omega_2; \nu) \leq \rho := 1 - \frac{\epsilon^4}{2}$. Apply Theorem 6 ($\rho \leftarrow \rho, \alpha \leftarrow \epsilon^2, \epsilon \leftarrow \frac{\Gamma_\rho(\frac{\epsilon}{3}, \frac{\epsilon}{3})}{2}$) to get τ and d. We will later apply this theorem with the parameters obtained here. Fix an arbitrary subset $C \subseteq V$, and $C_i := C \cap v^i$. For $0 \leq i \leq b$, call v^i *blocked* if $\mu^{\otimes R}[C_i(x)] \geq 1 - \epsilon$. At most $\lfloor \frac{\mathsf{wt}(C)}{1-\epsilon} \rfloor$ v^i's can be blocked. Let k' be the number of blocked v^i's, and $z = b + 1 - k'$ be the number of unblocked v^i's. Let $\{v^{i_1}, \ldots, v^{i_z}\}$ be the set of unblocked v^i's with $i_1 < i_2 < \cdots < i_z$.

For $1 \leq j \leq z$, let $S_j \subseteq v^{i_j}$ be such that $x \in S_j$ if there exists a path $(p_0 = s, p_1, \ldots, p_{j-1}, v_x^{i_j})$ such that each $p_{j'} \in v^{i_{j'}} \setminus C$ $(1 \leq j' < j)$. For $1 \leq j \leq z$, let $f_j : \Omega^R \mapsto [0, 1]$ be the indicator function of S_j.

We prove that if none of f_j reveals any *influential coordinate*, $\mu^{\otimes R}(S_z) > 0$. Since any path passing $v^{i_1}, \ldots, v^{i_z}$ (bypassing only blocked v^i's) uses short edges at least $b - 2k'$ times, so the length of the shortest path after removing C is at most $2 + (b - 2k') + 2ak'$.

▶ **Lemma 12.** *Suppose that for any $1 \leq j \leq z$ and $1 \leq i \leq R$, $\mathsf{Inf}_i^{\leq d}[f_j] \leq \tau$. Then $\mu^{\otimes R}(S_z) > 0$.*

Proof. We prove by induction that $\mu^{\otimes R}(S_j) \geq \frac{\epsilon}{3}$. It holds when $j = 1$ since v^{i_1} is unblocked. Assuming $\mu^{\otimes R}(S_j) \geq \frac{\epsilon}{3}$, since S_j does not reveal any influential coordinate, Theorem 6 shows that for any subset $T_{j+1} \subseteq v^{i_{j+1}}$ with $\mu^{\otimes R}(T_{j+1}) \geq \frac{\epsilon}{3}$, there exists an edge between S_j and T_{j+1}. If $S'_{j+1} \subseteq v^{i_{j+1}}$ is the set of neighbors of S_j, we have $\mu^{\otimes R}(S'_{j+1}) \geq 1 - \frac{\epsilon}{3}$. Since $v^{i_{j+1}}$ is unblocked, $\mu^{\otimes R}(S'_{j+1} \setminus C) \geq \frac{2\epsilon}{3}$, completing the induction. ◀

In summary, in the completeness case, if we cut vertices of total weight $k := k(a, b, r, \epsilon) = (b + 1)(\epsilon + \frac{1-\epsilon}{r})$, the length of the shortest path is at least $l := l(a, b, r, \epsilon) = a(b - r + 2)$. In the soundness case, even after cutting vertices of total weight k', the length of the shortest path is at most $2 + (b - \frac{k'}{1-\epsilon}) + 2a(\frac{k'}{1-\epsilon})$.

- Let $a = 4, b = 2r - 2$ and ϵ small enough so that $k \leq 2, l = 4r$. Requiring $l' \geq l$ results in $k' = \Omega(r)$, giving a gap of $\Omega(r) = \Omega(l)$ for Length Bounded Cut.
- Let $a = r, b = 2r - 2$ and ϵ small enough so that $k \leq 2, l = r^2$. Requiring $k' \leq 2$ results in $l' = O(r)$, giving a gap of $\Omega(r) = \Omega(\sqrt{l})$ for Shortest Path Interdiction. Generally, $k' \leq O(r^\epsilon)$ results in $l' \leq O(r^{1+\epsilon})$, giving an $(O(r^\epsilon), O(r^{1-\epsilon}))$-bicriteria gap for any $\epsilon \in (0, 1)$.

Acknowledgments. The author thanks Konstantin Makarychev for useful discussions on Directed Multicut, and Marek Elias for introducing Shortest Path Interdiction.

References

1. David Adjiashvili, Andrea Baggio, and Rico Zenklusen. Firefighting on trees beyond integrality gaps. *arXiv preprint arXiv:1601.00271*, 2016.
2. Amit Agarwal, Noga Alon, and Moses S. Charikar. Improved approximation for directed cut problems. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC'07, pages 671–680, New York, NY, USA, 2007. ACM. doi:10.1145/1250790.1250888.
3. Elliot Anshelevich, Deeparnab Chakrabarty, Ameya Hate, and Chaitanya Swamy. Approximability of the firefighter problem. *Algorithmica*, 62(1-2):520–536, 2012.
4. Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondřej Pangrác, Heiko Schilling, and Martin Skutella. Length-bounded cuts and flows. *ACM Transactions on Algorithms*, 7(1):4:1–4:27, December 2010. Preliminary version in ICALP'06. doi:10.1145/1868237.1868241.
5. Michael O Ball, Bruce L Golden, and Rakesh V Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8(2):73–76, 1989.
6. N. Bansal and S. Khot. Optimal long code test with one free bit. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'09, pages 453–462, Oct 2009. doi:10.1109/FOCS.2009.23.
7. Halil Bayrak and Matthew D Bailey. Shortest path network interdiction with asymmetric information. *Networks*, 52(3):133–140, 2008.
8. Cristina Bazgan, André Nichterlein, and Rolf Niedermeier. A refined complexity analysis of finding the most vital edges for undirected shortest paths. In Vangelis Th. Paschos and Peter Widmayer, editors, *Algorithms and Complexity*, volume 9079 of *Lecture Notes in Computer Science*, pages 47–60. Springer International Publishing, 2015. doi:10.1007/978-3-319-18173-8_3.
9. Parinya Chalermsook and Julia Chuzhoy. Resource minimization for fire containment. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'10, pages 1334–1349, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1873601.1873709.

10 Parinya Chalermsook and Daniel Vaz. New integrality gap results for the firefighters problem on trees. *arXiv preprint arXiv:1601.02388*, 2016.

11 Chandra Chekuri and Vivek Madan. Simple and fast rounding algorithms for directed and node-weighted multiway cut. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 797–807, 2016.

12 Chandra Chekuri and Vivek Madan. Approximating multicut and the demand graph. *arXiv preprint arXiv:1607.07200*, 2017. To appear in SODA'17.

13 Julia Chuzhoy and Sanjeev Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. *Journal of the ACM*, 56(2), 2009.

14 HW Corley and Y Sha David. Most vital links and nodes in weighted networks. *Operations Research Letters*, 1(4):157–160, 1982.

15 Pavel Dvořák and Dušan Knop. Parametrized complexity of length-bounded cuts and multicuts. In Rahul Jain, Sanjay Jain, and Frank Stephan, editors, *Theory and Applications of Models of Computation*, volume 9076 of *Lecture Notes in Computer Science*, pages 441–452. Springer International Publishing, 2015. doi:10.1007/978-3-319-17142-5_37.

16 Alina Ene, Jan Vondrák, and Yi Wu. Local distribution and the symmetry gap: Approximability of multiway partitioning problems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 306–325. SIAM, 2013.

17 Till Fluschnik, Danny Hermelin, André Nichterlein, and Rolf Niedermeier. Fractals for kernelization lower bounds, with an application to length-bounded cut problems. *arXiv preprint arXiv:1512.00333*, 2015.

18 Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. Multiway cuts in directed and node weighted graphs. In *Automata, Languages and Programming*, pages 487–498. Springer, 1994.

19 Petr A. Golovach and Dimitrios M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72 – 86, 2011. Parameterized Complexity of Discrete Optimization. doi:http://dx.doi.org/10.1016/j.disopt.2010.09.009.

20 Venkatesan Guruswami and Euiwoong Lee. Simple proof of hardness of feedback vertex set. *Theory of Computing*, 12(6):1–11, 2016.

21 Venkatesan Guruswami, Sushant Sachdeva, and Rishi Saket. Inapproximability of minimum vertex cover on k-uniform k-partite hypergraphs. *SIAM Journal on Discrete Mathematics*, 29(1):36–58, 2015.

22 Bert Hartnell. Firefighter! an application of domination. presentation. In *25th Manitoba Conference on Combinatorial Mathematics and Computing, University of Manitoba in Winnipeg, Canada*, 1995.

23 Eitan Israeli and R Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

24 Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir Gurvich, Gabor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2007. doi:10.1007/s00224-007-9025-6.

25 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing*, STOC'02, pages 767–775, 2002.

26 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for Max-Cut and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

27 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

28 Andrew King and Gary MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.

29 Amit Kumar, Rajsekar Manokaran, Madhur Tulsiani, and Nisheeth K. Vishnoi. On LP-based approximability for strict CSPs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'11, pages 1560–1573. SIAM, 2011. URL: `http://dl.acm.org/citation.cfm?id=2133036.2133157`.

30 Euiwoong Lee. Improved hardness for cut, interdiction, and firefighter problems. *arXiv preprint arXiv:1607.05133*, 2016. URL: `https://arxiv.org/abs/1607.05133`

31 László Lovász, V Neumann-Lara, and M Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9(4):269–276, 1978.

32 A Ridha Mahjoub and S Thomas McCormick. Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. *Mathematical programming*, 124(1-2):271–284, 2010.

33 K. Malik, A. K. Mittal, and S. K. Gupta. The k most vital arcs in the shortest path problem. *Oper. Res. Lett.*, 8(4):223–227, August 1989. `doi:10.1016/0167-6377(89)90065-5`.

34 Rajsekar Manokaran, Joseph Seffi Naor, Prasad Raghavendra, and Roy Schwartz. Sdp gaps and ugc hardness for multiway cut, 0-extension, and metric labeling. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 11–20. ACM, 2008.

35 David P Morton. Stochastic network interdiction. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

36 Elchanan Mossel. Gaussian bounds for noise correlation of functions. *Geometric and Functional Analysis*, 19(6):1713–1756, 2010. `doi:10.1007/s00039-010-0047-x`.

37 Joseph Naor and Leonid Zosin. A 2-approximation algorithm for the directed multiway cut problem. *SIAM Journal on Computing*, 31(2):477–482, 2001.

38 K. Okumoto, T. Fukunaga, and H. Nagamochi. Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems. *Algorithmica*, 62(3):787–806, 2012.

39 Feng Pan and Aaron Schild. Interdiction problems on planar graphs. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and JoséD.P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 8096 of *Lecture Notes in Computer Science*, pages 317–331. Springer Berlin Heidelberg, 2013. `doi:10.1007/978-3-642-40328-6_23`.

40 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC'08, pages 245–254, 2008.

41 Michael Saks, Alex Samorodnitsky, and Leonid Zosin. A lower bound on the integrality gap for minimum multicut in directed networks. *Combinatorica*, 24(3):525–530, 2004.

42 J Cole Smith, Mike Prince, and Joseph Geunes. Modern network interdiction problems and algorithms. In *Handbook of Combinatorial Optimization*, pages 1949–1987. Springer, 2013.

43 Ola Svensson. Hardness of vertex deletion and project scheduling. *Theory of Computing*, 9(24):759–781, 2013. Preliminary version in APPROX'12. `doi:10.4086/toc.2013.v009a024`.

44 Cenny Wenner. Circumventing d-to-1 for approximation resistance of satisfiable predicates strictly containing parity of width four. *Theory of Computing*, 9(23):703–757, 2013.

Subspace-Invariant AC^0 Formulas

Benjamin Rossman*

University of Toronto, Toronto, Canada
rossman@utoronto.ca

Abstract

The n-variable PARITY function is computable (by a well-known recursive construction) by AC^0 formulas of depth $d+1$ and leafsize $n \cdot 2^{dn^{1/d}}$. These formulas are seen to possess a certain symmetry: they are syntactically invariant under the subspace P of even-weight elements in $\{0,1\}^n$, which acts (as a group) on formulas by toggling negations on input literals. In this paper, we prove a $2^{d(n^{1/d}-1)}$ lower bound on the size of syntactically P-invariant depth $d+1$ formulas for PARITY. Quantitatively, this beats the best $2^{\Omega(d(n^{1/d}-1))}$ lower bound in the non-invariant setting [16].

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases lower bounds, size-depth tradeoff, parity, symmetry in computation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.93

1 Introduction

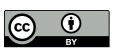
Let U be a linear subspace of $\{0,1\}^n$. We say that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *U-invariant* if $f(x) = f(x \oplus u)$ for all $u \in U$ and $x \in \{0,1\}^n$. (Note that U-invariant Boolean functions are in one-to-one correspondence with functions from the quotient space $\{0,1\}^n/U$ to $\{0,1\}$.) An obvious example is the PARITY function $x \mapsto \bigoplus_{i=1}^n x$, which is P-invariant where P is the linear subspace of even-weight elements in $\{0,1\}^n$.

We may also view U as a group that acts on the set of n-variable Boolean circuits (as well as the set of n-variable Boolean formulas). Here we consider circuits with unbounded fan-in AND and OR gates and inputs labeled by literals in the set $\{X_1, \overline{X}_1, \ldots, X_n, \overline{X}_n\}$, also known as *$AC^0$ circuits* in the setting where depth is bounded. For a circuit C and an element $u \in U$, let C^u be the circuit obtained from C by negating the ith pair of literals (i.e. exchanging X_i and $\overline{X}_i$ as labels on inputs) for all coordinates $i \in [n]$ such that $u_i = 1$. This action of U on circuits is compatible with the action on Boolean functions: for all $u \in U$ and $x \in \{0,1\}^n$, we have $C^u(x) = C(x \oplus u)$.

There are two notions of U-invariance for circuits. We say that C is *syntactically U-invariant* if C is identical to C^u for every $u \in U$ (we define this notion precisely for formulas), while we say that C is *semantically U-invariant* if it computes a U-invariant function. Syntactic U-invariance clearly implies semantic U-invariance. However, the converse is false: a circuit may compute a U-invariant function without being syntactically U-invariant.

For a U-invariant Boolean function f, we define its *U-invariant circuit size* as the minimum number of gates in a syntactically U-invariant circuit that computes it. This quantity may be compared to the usual ("non-invariant") circuit size of f. There are several questions we may ask: What gap, if any, is there between the U-invariant circuit size and non-invariant circuit size of f? Are lower bounds for U-invariant circuit size easier to prove,

* Supported by NSERC.



44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 93; pp. 93:1–93:11

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

and do they suggest new strategies for proving lower bound in the non-invariant setting? The same questions may be asked with respect to U-invariant versions of other complexity measures, such as formula size and bounded-depth versions of both circuit and formula size (noting that the action of U on circuits preserves fan-out and depth).[1]

In this paper, we focus on bounded-depth formula size. Our primary target is the P-invariant PARITY function where P is the linear subspace of even-weight elements in $\{0,1\}^n$. We start from the observation that the best known construction of bounded-depth circuits and formulas for PARITY are syntactically P-invariant. Here we refer to the well-known recursive construction, for all $d \geq 1$, of depth $d+1$ circuits and formulas for PARITY, of size at most $n \cdot 2^{n^{1/d}}$ and $n \cdot 2^{dn^{1/d}}$ respectively. The main result of this paper (Theorem 1) yields a nearly matching lower bound of $2^{d(n^{1/d}-1)}$ on the P-invariant depth $d+1$ formula size of PARITY. This implies a $2^{n^{1/d}-1}$ lower bound on P-invariant depth $d+1$ circuit size.[2] Quantitatively, the lower bounds are stronger than the best known $\Omega(2^{\frac{1}{10}n^{1/d}})$ and $\Omega(2^{\frac{1}{84}d(n^{1/d}-1)})$ lower bounds for non-invariant depth $d+1$ circuits [10] and formulas [16], respectively. Qualitatively, syntactic P-invariance appears to be a severe restriction and unnatural from the standpoint of computation.

The general form of our lower bound is the following theorem.

▶ **Theorem 1.** *Let $U \subset V$ be linear subspaces of $\{0,1\}^n$, and suppose F is a syntactically U-invariant depth $d+1$ formula which is non-constant over V. Then F has size at least $2^{d(m^{1/d}-1)}$ where $m = \min\{|x| : x \in U^\perp \setminus V^\perp\}$ (i.e. m is the minimum Hamming weight of a vector x which is orthogonal to U but non-orthogonal to V).*

Some observations: first, notice that the bound in Theorem 1 does not depend on the parameter n, i.e. the dimension of the ambient hypercube. The lower bound for PARITY described in the previous paragraph is the special case $U = P$ and $V = \{0,1\}^n$. Theorem 1 implies an $m^{1/\log_2(e)}$ lower bound for unbounded-depth formulas, since $\lim_{d\to\infty} d(m^{1/d}-1) = \ln(m)$. It also implies a $2^{m^{1/d}-1}$ lower bound for depth $d+1$ circuits. (However, we get no non-trivial lower bound for unbounded-depth circuits, since $\lim_{d\to\infty} m^{1/d} - 1 = 0$.)

The proof of Theorem 1 uses elementary linear algebra, in particular a small lemma on the existence of linear retractions with small Hamming-weight distortion (Lemma 5). Overall, this is much simpler than the random restriction and polynomial approximation methods typically used to prove AC^0 lower bounds.

1.1 Related Work

Syntactically invariant models of computation have been previously studied from the perspective of Descriptive Complexity, an area that characterizes complexity classes in terms of definability in different logics [11]. In this context, the notion of invariance pertains to the action of S_m on $n = \binom{m}{2}$ binary variables, encoding the edge relation of a simple graph on m vertices. More generally, for a finite relational signature σ, one may consider the action of S_m on $n = \sum_{R\in\sigma} m^{\text{arity}(R)}$ binary variables (encoding the possible σ-structures with universe $\{1,\ldots,m\}$). The action of S_m on the set of variables $\{X_1,\ldots,X_n\}$ induces a

[1] These questions have been asked previously concerning, e.g., the action of the symmetric group S_n on n-invariable circuits. For S_n-invariant Boolean functions (a.k.a. symmetric functions) including PARITY and MAJORITY, there is known to be an exponential gap between U-invariant and non-invariant circuit and formula size. (See the Related Work section, below.)

[2] This follows from the observation that every [syntactically U-invariant] depth $d+1$ circuit of size s is equivalent to a [syntactically U-invariant] depth $d+1$ formula of size at most s^d.

syntactic action of S_m on the set of n-variable Boolean circuits (and many other concrete models of computation, such as branching programs, etc.)

An early result in this area, due to Denenberg et al [8], shows that syntactically S_m-invariant circuits of polynomial size and constant depth (subject to a certain uniformity condition) capture precisely the first-order definable properties of finite σ-structures. A decade later, Otto [13] introduced a certain limit object of finite circuits (also viewed as a form of uniformity) and showed a correspondence between infinitary logic with a bounded number of variables ($L^{\omega}_{\infty\omega}$) and syntactically S_m-invariant circuits of polynomial size and arbitrary depth. Otto also gives characterizations of fixed point and partial fixed point logic in terms of syntactically S_m-invariant networks. More recently, Anderson and Dawar [2] showed a correspondence (under a different uniformity condition) between fixed-point logic (FP) and syntactically S_m-invariant polynomial-size circuits, as well between fixed-point logic with counting (FPC) and syntactically S_m-invariant polynomial-size circuits in the basis that includes majority gates.

So far as I know, this paper is the first to study syntactic invariance under the action of linear subspaces of $\{0,1\}^n$ (i.e. subgroups on $\mathbb{Z}_2^n$) on n-variable Boolean circuits. A different notion of syntactic invariance — with respect to the automorphism group of the input structure — can be found in the literature on Choiceless Polynomial Time [3, 4, 6, 7, 9, 15]. On S_m-invariant tautologies in proof complexity, see [1, 14].

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \dots\}$. Let n and d be arbitrary positive integers. Let $[n] = \{1, \dots, n\}$.

Our lower bound makes use of the following inequality involving the function $n \mapsto dn^{1/d}$:

▶ **Lemma 2.** *For all real numbers $a, b, c > 0$, we have*

$$a + c(b/a)^{1/c} \geq (c+1)b^{1/(c+1)}$$

with equality iff $a = b^{1/(c+1)}$.

Proof. We have $\frac{\partial}{\partial a}\big(a + c(b/a)^{1/c}\big) = 1 - (b/a^{(c+1)})^{1/c}$. Thus, the function $a \mapsto a + c(b/a)^{1/c}$ is seen to have a unique minimum at $a = b^{1/(c+1)}$ where it takes value $(c+1)b^{1/(c+1)}$. ◀

2.1 Linear Algebra

For $x, y \in \{0,1\}^n$, we write $|x| := \sum_{i=1}^n x_i$ for the Hamming weight of x, we write $x \oplus y$ for the bitwise sum of x and y modulo 2 (i.e. the element $z \in \{0,1\}^n$ with $z_i := x_i \oplus y_i$), and we write $\langle x, y\rangle := \bigoplus_{i=1}^n x_i y_i$ for the inner product of x and y.

We write $\mathcal{L}$ for the lattice of linear subspaces of $\{0,1\}^n$. For $U, V \in \mathcal{L}$, we write $\dim(V)$ for the dimension of V, we write $V^{\perp} := \{x \in \{0,1\}^n : \langle x, v\rangle = 0 \text{ for all } v \in V\}$ for the orthogonal complement of V, and we write $U + V$ for the subspace spanned by U and V. We say that U is a *codimension-k subspace* of V if $U \subseteq V$ and $\dim(V) - \dim(U) = k$.

The orthogonal complement has the following properties:

$$\dim(V) + \dim(V^{\perp}) = n, \qquad U \subseteq V \iff V^{\perp} \subseteq U^{\perp},$$
$$V = (V^{\perp})^{\perp}, \qquad (U+V)^{\perp} = U^{\perp} \cap V^{\perp}, \qquad (U \cap V)^{\perp} = U^{\perp} + V^{\perp}.$$

2.2 AC^0 Formulas

We write $\mathcal{F}$ for the set of n-variable AC^0 formulas (with unbounded fan-in AND and OR gates and leaves labeled by literals). Formally, let $\mathcal{F} = \bigcup_{d\in\mathbb{N}} \mathcal{F}_d$ where $\mathcal{F}_d$ is the set of *depth-d formulas*, defined inductively as follows:[3]

- $\mathcal{F}_0$ is the set of literals $\{X_1, \ldots, X_n, \overline{X}_1, \ldots, \overline{X}_n\}$,
- $\mathcal{F}_{d+1}$ is the set of ordered pairs $\{(\gamma, \mathcal{G}) : \gamma \in \{\text{AND}, \text{OR}\}$ and $\mathcal{G}$ is a nonempty subset of $\mathcal{F}_d\}$.

Every $F \in \mathcal{F}$ computes a Boolean function $\{0,1\}^n \to \{0,1\}$, defined in the usual way. For $x \in \{0,1\}^n$, we write $F(x)$ for the value of F on x.

For a nonempty set $S \subseteq \{0,1\}^n$ and $b \in \{0,1\}$, notation $F(S) \equiv b$ is the assertion that $F(x) = b$ for all $x \in S$. We say that F is *non-constant* on S if $F(S) \not\equiv 0$ and $F(S) \not\equiv 1$ (i.e. there exist $x, y \in S$ such that $F(x) = 0$ and $F(y) = 1$).

The *depth* of F is the minimum d such that $F \in \mathcal{F}_d$. The *leafsize* of a formula is the number of depth-0 subformulas. Let *size* of a formula refer to the number of depth-1 subformulas. Inductively,

$$\text{leafsize}(F) = \begin{cases} 0 & \text{if } F \in \mathcal{F}_0, \\ \sum_{G\in\mathcal{G}} \text{size}(G) & \text{if } F = (\gamma, \mathcal{G}) \in \mathcal{F} \setminus \mathcal{F}_0, \end{cases}$$

$$\text{size}(F) = \begin{cases} 0 & \text{if } F \in \mathcal{F}_0, \\ 1 & \text{if } F \in \mathcal{F}_1, \\ \sum_{G\in\mathcal{G}} \text{size}(G) & \text{if } F = (\gamma, \mathcal{G}) \in \mathcal{F} \setminus (\mathcal{F}_0 \cup \mathcal{F}_1). \end{cases}$$

Clearly $\text{size}(F) \le \text{leafsize}(F)$. (Note that size is within a factor 2 of the number of gates in F, which is how one usually measures size of circuits.) We define these two complexity measures since our lower bound naturally applies to size, while the upper bounds are naturally stated in terms of leafsize.

2.3 The Action of $\{0,1\}^n$

We define a group action of $\{0,1\}^n$ on $\mathcal{F}$ as follows. For $u \in \{0,1\}^n$ and $F \in \mathcal{F}$, let F^u be the formula obtained from F by exchanging literals X_i and $\overline{X}_i$ for every $i \in [n]$ with $u_i = 1$. Formally, this action is defined inductively by

$$F^u = \begin{cases} X_i \text{ (resp. } \overline{X}_i) & \text{if } F = X_i \text{ (resp. } \overline{X}_i) \text{ and } u_i = 0, \\ \overline{X}_i \text{ (resp. } X_i) & \text{if } F = X_i \text{ (resp. } \overline{X}_i) \text{ and } u_i = 1, \\ (\gamma, \{G^u : G \in \mathcal{G}\}) & \text{if } F = (\gamma, \mathcal{G}). \end{cases}$$

Clearly F^u has the same depth and size as F. Note that $F^u(x) = F(x \oplus u)$ for all $x \in \{0,1\}^n$.

If U is a linear subspace of $\{0,1\}$ (i.e. subgroup of $\{0,1\}^n$), then we say that an AC^0 formula F is:

- *syntactically U-invariant* if $F^u = F$ for every $u \in U$,
- *semantically U-invariant* if $F(x) = F(x \oplus u)$ for every $u \in U$ and $x \in \{0,1\}^n$.

As remarked in Section 1, syntactic U-invariance implies semantic U-invariance (but not conversely).

---

[3] As a minor convenience, we do not include constants 0 and 1 in $\mathcal{F}_0$, nor do we allow identical sibling subformulas (i.e. multisets $\mathcal{G}$) in the definition of $\mathcal{F}_{d+1}$. This is without loss of generality: the *depth-d formula size* of a Boolean function is unaffected by these restrictions.

2.4 Upper Bound

We briefly review the smallest known construction of bounded-depth formulas for PARITY and observe that these formulas are syntactically P-invariant.

▶ **Proposition 3.** *For all $d, n \geq 1$, the n-variable* PARITY *function is computable by syntactically P-invariant depth $d+1$ formulas of leafsize at most $n \cdot 2^{dn^{1/d}}$ where P is the even-weight subspace of $\{0,1\}^n$. If $n^{1/d}$ is an integer, this bound improves to $n \cdot 2^{d(n^{1/d}-1)}$.*

Proof. For an optimal choice of $k, n_1, \ldots, n_k \geq 1$ with $n_1 + \cdots + n_k = n$, we construct a syntactically P_n-invariant depth $d+1$ formula for PARITY_n — with output gate OR (resp. AND) — by composing the brute-force DNF (resp. CNF) for PARITY_k (in which each variable occurs 2^{k-1} times) with syntactically P_{n_i}-invariant depth d formulas for PARITY_{n_i} (or $1 - \text{PARITY}_{n_i}$) with output gate AND (resp. OR). The minimum leafsize $\beta(d+1, n)$ achievable by this construction is given by the recurrence

$$\beta(1,n) = \begin{cases} 1 & \text{if } n = 1, \\ \infty & \text{if } n > 1, \end{cases} \qquad \beta(d+1,n) = \min_{\substack{k, n_1, \ldots, n_k \geq 1 \\ n_1 + \cdots + n_k = n}} 2^{k-1} \sum_{i=1}^{k} \beta(d, n_i).$$

We now observe:

- If $n^{1/d}$ is an integer, we get $\beta(d+1,n) \leq n \cdot 2^{d(n^{1/d}-1)}$ by setting $k = n^{1/d}$ and $n_1 = \cdots = n_k = n^{(d-1)/d}$.
- For arbitrary $d, n \geq 1$, we get $\beta(d+1,n) \leq n \cdot 2^{dn^{1/d}}$ by setting $k = \lceil n/t \rceil$ and $n_1, \ldots, n_k \in \{t-1, t\}$ where $t = \lfloor n^{(d-1)/d} \rfloor$. ◀

An aside

I suspect that, by analyzing the above recurrence more carefully, the upper bound in Proposition 3 can be improved to $O(n \cdot 2^{d(n^{1/d}-1)})$ for all $d \leq \lceil \log n \rceil$. This is suggested by the observation that PARITY is computable by syntactically P-invariant formulas of depth $\lceil \log n \rceil + 1$ and leafsize $O(n^2)$. Note that the upper bound of Proposition 3 is slack (except when $n^{1/d}$ is an integer), since setting $d = \log n$, we have $n \cdot 2^{d(n^{1/d}-1)} = n^2$ and $n \cdot 2^{dn^{1/d}} = n^3$. Also note that $O(n \cdot 2^{d(n^{1/d}-1)})$ is *not* an upper bound for $d \gg \log n$, since $\Omega(n^2)$ is lower bound even for non-invariant formulas of unbounded depth [12].

3 Linear-Algebraic Lemmas

In this section, we prove a linear-algebraic lemma (Lemma 9) which plays a key role in our lower bound. Recall that S, T, U, V range over the set of linear subspaces of $\{0,1\}^n$, denoted by $\mathcal{L}$.

▶ **Definition 4.** For linear spaces $U \subseteq V$, a *linear retraction* from V to U is a linear function $\rho : V \to U$ such that $\rho(u) = u$ for every $u \in U$.

We next give a small lemma on the existence of linear retractions with small (one-sided) Hamming-weight distortion.

▶ **Lemma 5.** *If U is a codimension-k subspace of V, then there exists a linear retraction $\rho : V \to U$ such that $|\rho(v)|/|v| \leq k+1$ for all $v \in V$.*

Proof. Greedily choose a basis $w_1, \ldots, w_k$ for V over U such that w_i has minimal Hamming weight among elements of $V \setminus \text{Span}(U \cup \{w_1, \ldots, w_{i-1}\})$ for all $i \in [k]$. Each $v \in V$ has a

unique representation $v = u \oplus a_1 w_1 \oplus \cdots \oplus a_k w_k$ where $u \in U$ and $a_1, \ldots, a_k \in \{0,1\}$. Let $\rho : V \to U$ be the map $v \mapsto u$ and observe that this is a linear retraction.

To show that $|\rho(v)| \leq (k+1)|v|$, we first notice that $|a_i w_i| \leq |v|$ for all $i \in [k]$. If $a_i = 0$, this is obvious, as $|a_i w_i| = 0$. If $a_i = 1$, then $v \in V \setminus \mathrm{Span}(U \cup \{w_1, \ldots, w_{i-1}\})$, so by our choice of w_i we have $|a_i w_i| = |w_i| \leq |v|$. Completing the proof, we have

$$\begin{aligned} |\rho(v)| &= |v \oplus a_1 v_1 \oplus \cdots \oplus a_k v_k| \\ &\leq |v| + |a_1 v_1| + \cdots + |a_k v_k| \\ &\leq (k+1)|v|. \end{aligned}$$

◀

▶ **Definition 6.** Define sets $\mathcal{L}_2$ and $\mathcal{L}_4$ as follows:

$$\mathcal{L}_2 = \big\{(U,V) \in \mathcal{L} \times \mathcal{L} : U \text{ is a codimension-1 subspace of } V\big\},$$

$$\mathcal{L}_4 = \big\{((S,T),(U,V)) \in \mathcal{L}_2 \times \mathcal{L}_2 : T \cap U = S \text{ and } T + U = V\big\}.$$

The next lemma shows that $\mathcal{L}_4$ is symmetric under orthogonal complementation.

▶ **Lemma 7.** *For all $((S,T),(U,V)) \in \mathcal{L}_4$, we have $((V^\perp, U^\perp), (T^\perp, S^\perp)) \in \mathcal{L}_4$.*

Proof. This follows from the properties of the orthogonal complement listed in §2.1. Consider any $((S,T),(U,V)) \in \mathcal{L}_4$. First note that $(V^\perp, U^\perp) \in \mathcal{L}_2$ by the fact that $U \subseteq V \implies V^\perp \subseteq U^\perp$ and $\dim(U^\perp) - \dim(V^\perp) = (n - \dim(U)) - (n - \dim(V)) = \dim(V) - \dim(U) = 1$. Similarly, we have $(T^\perp, S^\perp) \in \mathcal{L}_2$. We now have $((V^\perp, U^\perp), (T^\perp, S^\perp)) \in \mathcal{L}_4$ since $U^\perp \cap T^\perp = (T+U)^\perp = V^\perp$ and $U^\perp + T^\perp = (T \cap U)^\perp = S^\perp$. ◀

▶ **Lemma 8.** *For all $S \subset T \subseteq V$ such that $(S,T) \in \mathcal{L}_2$, there exists $U \supseteq S$ such that $((S,T),(U,V)) \in \mathcal{L}_4$ and*

$$\min_{x \in V \setminus U} |x| \geq \frac{1}{\dim(V) - \dim(T) + 1} \min_{y \in T \setminus S} |y|.$$

Proof. By Lemma 5, there exists a linear retraction $\rho : V \to T$ such that $|\rho(v)|/|v| \leq \dim(V) - \dim(T) + 1$ for all $v \in V$. Let $U = \rho^{-1}(S)$ and note that U is a codimension-1 subspace of V. (This follows from applying the Rank-Nullity Theorem to linear functions $\rho : V \to T$ and $\rho|U : U \to S$ and noting that $\ker(\rho) = \ker(\rho|U)$.) We have $S = T \cap U$ and $T + U = V$, hence $((S,T),(U,V)) \in \mathcal{L}_4$. Choosing x with minimum Hamming weight in $V \setminus U$, we observe that $\rho(x) \in T \setminus S$ and $|x| \geq |\rho(v)|/(\dim(V) - \dim(T) + 1)$, which proves the lemma. ◀

▶ **Lemma 9.** *For all $S \subseteq U \subset V$ such that $(U,V) \in \mathcal{L}_2$, there exists $T \subseteq V$ such that $((S,T),(U,V)) \in \mathcal{L}_4$ and*

$$\min_{x \in S^\perp \setminus T^\perp} |x| \geq \frac{1}{\dim(U) - \dim(S) + 1} \min_{y \in U^\perp \setminus V^\perp} |y|.$$

Proof. Follows directly from Lemmas 7 and 8. ◀

4 Proof of Theorem 1

The following lemma gives the base case of Theorem 1 for depth-2 formulas (a.k.a. DNFs and CNFs). In this case, we merely require the hypothesis of semantic rather than syntactic U-invariance. The proof is similar to the standard argument showing that depth-2 formulas for PARITY require 2^{n-1} clauses of width n.

▶ **Lemma 10.** *Suppose F is a depth-2 formula and $(U,V) \in \mathcal{L}_2$ such that $F(U) \equiv b$ and $F(V \setminus U) \equiv 1-b$ for some $b \in \{0,1\}$. Then* $\text{size}(F) \geq 2^{m-1}$ *and* $\text{leafsize}(F) \geq m \cdot 2^{m-1}$ *where* $m = \min\{|x| : x \in U^\perp \setminus V^\perp\}$.

Proof. Without loss of generality, assume that F is a DNF formula (i.e. an OR-of-ANDs formula) and $F(U) \equiv 0$ and $F(V \setminus U) \equiv 1$. (The argument is similar if we replace DNF with CNF, or if we assume that $F(U) \equiv 1$ and $F(V \setminus U) \equiv 0$.) We further assume that F is minimal with respect to the number of clauses and the number of literals in any particular clause.

Consider a clause G of F. This clause G is the AND of some number ℓ of literals. Without loss of generality, suppose these literals involve the first ℓ coordinates. Let π be the projection map $\{0,1\}^n \to \{0,1\}^\ell$. Then there is a point $p \in \{0,1\}^\ell$ such that $G(x) = 1 \iff \pi(x) = p$ for all $x \in \{0,1\}^n$. Observe that $G(U) \equiv 0$ (since $F(U) \equiv 0$) and, therefore, $p \notin \pi(U)$.

We claim that $p \in \pi(V \setminus U)$. To see why, assume for contradiction that $p \notin \pi(V \setminus U)$. Then $G(V) \equiv 0$. But this means that the clause G can be removed from F and the resulting function would still satisfy $F(U) \equiv 0$ and $F(V \setminus U) \equiv 1$. This contradicts the minimality of F with respect to number of clauses.

For each $i \in [\ell]$, let $p^{(i)}$ be the neighbor of p in $\{0,1\}^\ell$ along the ith coordinate. We claim that $p^{(1)}, \ldots, p^{(\ell)} \in \pi(U)$. Without loss of generality, we give the argument showing $p^{(\ell)} \in \pi(U)$. Let G' be the AND of the first $\ell - 1$ literals in G, and let F' be the formula obtained from F by replacing G with G'. For all $x \in \{0,1\}^n$, we have $G(x) \leq G'(x)$ and hence $F(x) \leq F'(x)$. Therefore, $F'(V \setminus U) \equiv 1$. We now note that there exists $u \in U$ such that $F'(u) = 1$ (otherwise, we would have $F'(u) \equiv 0$, contradicting the minimality of F with respect to the width of each clause). Since $F(u) = 0$ and G' is the only clause of F' distinct from the clauses of F, it follows that $G'(u) = 1$. This means that $u_{\{1,\ldots,\ell-1\}} = p_{\{1,\ldots,\ell-1\}}$. We now have $\pi(u) = p^{(\ell)}$ (otherwise, we would have $\pi(u) = p$ and therefore $G(u) = 1$ and $F(u) = 1$, contradicting that fact that $F(U) \equiv 0$).

Since π is a linear function and $\pi(U) \neq \pi(V)$, it follows that $\pi(U)$ is a codimension-1 subspace of $\pi(V)$. The fact that $p \in \pi(V \setminus U)$ and $p^{(1)}, \ldots, p^{(\ell)} \in \pi(U)$ now forces $\pi(V) = \{0,1\}^\ell$ and $\pi(U) = \{q \in \{0,1\}^\ell : |q| \text{ is even}\}$. Therefore, $1^\ell \in \pi(U)^\perp \setminus \pi(V)^\perp$ (writing 1^ℓ for the all-1 vector in $\{0,1\}^\ell$). It follows that $1^\ell 0^{n-\ell} \in U^\perp \setminus V^\perp$ and, therefore, $\ell = |1^\ell 0^{n-\ell}| \geq m$ (by definition of m).

We now observe that

$$\mathop{\mathbb{P}}_{v \in V}[G(v) = 1] = \mathop{\mathbb{P}}_{v \in V}[\pi(v) = p] = \mathop{\mathbb{P}}_{q \in \pi(V)}[q = p] = \mathop{\mathbb{P}}_{q \in \{0,1\}^\ell}[q = p] = 2^{-\ell} \leq 2^{-m}.$$

That is, each clause in F has value 1 over at most 2^{-m} fraction of points in V. Since the set $V \setminus U$ has density $1/2$ in V, we see that 2^{m-1} clauses are required to cover $V \setminus U$.

Subject to the stated minimality assumptions on F (with respect to the number of clauses and, secondarily, to the width of each clause), we conclude that F contains $\geq 2^{m-1}$ clauses, each of width $\geq m$. Therefore, $\text{size}(F) \geq 2^{m-1}$ and $\text{leafsize}(F) \geq m \cdot 2^{m-1}$. ◀

On to our main result:

▶ **Theorem 1 (restated).** *Let $U \subset V$ be linear subspaces of $\{0,1\}^n$, and suppose F is a syntactically U-invariant depth $d+1$ formula which is non-constant over V. Then F has size at least $2^{d(m^{1/d}-1)}$ where $m = \min\{|x| : x \in U^\perp \setminus V^\perp\}$.*

Proof. We first observe that it suffices to prove the theorem in the case where $(U,V) \in \mathcal{L}_2$, that is, U has codimension-1 in V. To see why, note that for any $U \subset V$ where F is

syntactically U-invariant and non-constant over V, there must exist $U \subset W \subseteq V$ such that $(U, W) \in \mathcal{L}_2$ and F is non-constant over W. Assuming the theorem holds with respect to $U \subset W$, it also hold with respect to $U \subset V$, since $U^\perp \setminus V^\perp \subseteq U^\perp \setminus W^\perp$ and hence $\min\{|x| : x \in U^\perp \setminus V^\perp\} \geq \min\{|x| : x \in U^\perp \setminus W^\perp\}$.

Therefore, we assume $(U, V) \in \mathcal{L}_2$ and prove the theorem by induction on d. The base case $d = 1$ is established by Lemma 10.[4] For the induction step, let $d \geq 2$ and assume $F \in \mathcal{F}_{d+1}$ is a syntactically U-invariant and non-constant over V. Without loss of generality, we consider the case where $F = (\mathrm{OR}, \mathcal{G})$ for some nonempty $\mathcal{G} \subseteq \mathcal{F}_d$. (The case where $F = (\mathrm{AND}, \mathcal{G})$ is symmetric, with the roles of 0 and 1 exchanged.)

Since F is syntactically U-invariant, we have $G^u \in \mathcal{G}$ for every $u \in U$ and $G \in \mathcal{G}$. We claim that it suffices to prove the theorem in the case where the action of U on $\mathcal{G}$ is transitive (i.e. $\mathcal{G} = \{G^u : u \in U\}$ for every $G \in \mathcal{G}$). To see why, consider the partition $\mathcal{G} = \mathcal{G}_1 \sqcup \cdots \sqcup \mathcal{G}_t$, $t \geq 1$, into orbits under U. For each $i \in [t]$, let F_i be the formula $(\mathrm{OR}, \mathcal{G}_i)$. Note that F_i is syntactically U-invariant and U acts transitively on $\mathcal{G}_i$. Clearly, we have $F(v) = \bigvee_{i \in [t]} F_i(v)$ for all $v \in V$. Since every U-invariant Boolean function is constant over sets U and $V \setminus U$ (using the fact that U has codimension-1 in V), this means that each F_i satisfies either $F_i(V) \equiv 0$ or $F(v) = F_i(v)$ for all $v \in V$. Because F is non-constant over V, it follows that there exists $i \in [t]$ such that $F(v) = F_i(v)$ for all $v \in V$. In particular, this F_i is non-constant over V. Since $\mathrm{size}(F) \geq \mathrm{size}(F_i)$, we have reduced proving the theorem for F to proving to theorem for F_i.

In light of the preceding paragraph, we proceed under the assumption that U acts transitively on $\mathcal{G}$. Fix an arbitrary choice of $G \in \mathcal{G}$. Let

$$S = \mathrm{Stab}_U(G) \ (= \{u \in U : G^u = G\}),$$
$$a = \dim(U) - \dim(S) + 1.$$

By the Orbit-Stabilizer Theorem,

$$|\mathcal{G}| = |\mathrm{Orbit}_U(G)| = [U : S] = |U|/|S| = 2^{a-1}.$$

Since $\mathrm{size}(G') = \mathrm{size}(G)$ for every $G' \in \mathcal{G}$, we have

$$\mathrm{size}(F) = \sum_{G' \in \mathcal{G}} \mathrm{size}(G') = |\mathcal{G}| \cdot \mathrm{size}(G) = 2^{a-1} \cdot \mathrm{size}(G). \tag{1}$$

We next observe that G^u is syntactically S-invariant for every $u \in U$ (in fact, $S = \mathrm{Stab}_U(G^u)$). This follows from the fact that $(G^u)^s = G^{u \oplus s} = (G^s)^u = G^u$ for every $s \in S$.

By Lemma 9, there exists T such that $((S, T), (U, V)) \in \mathcal{L}_4$ and

$$\min_{x \in S^\perp \setminus T^\perp} |x| \geq \frac{1}{\dim(U) - \dim(S) + 1} \min_{y \in U^\perp \setminus V^\perp} |y| = \frac{m}{a}.$$

We claim that there exists $u \in U$ such that G^u is non-constant on T. There are two cases to consider:

- **Case 1:** *Suppose $F(U) \equiv 0$ and $F(V \setminus U) \equiv 1$.*
  We have $G(U) \equiv 0$ and $G(V) \not\equiv 0$. Fix any $v \in V \setminus U$ such that $G(v) = 1$. In addition, fix any $w \in T \setminus U$ (noting that $T \setminus U$ is nonempty since $U + T = V$ and $U \subset V$). Let $u = v \oplus w$ and note that $u \in U$ (since U is a codimension-1 subspace of V and $v, w \in V \setminus U$). We have $G^u(U) \equiv 0$ and $G^u(w) = G(w \oplus u) = G(v) = 1$. By the S-invariance of G^u, it follows that $G^u(S) \equiv 0$ and $G^u(T \setminus S) \equiv 1$. In particular, G^u is non-constant on T.

[4] Note that Theorem 1 makes sense even when $d = 0$, if we interpret $0 \cdot (m^{1/0} - 1)$ as 0 if $m = 1$ and ∞ if $m > 1$.

- **Case 2:** *Suppose* $F(U) \equiv 1$ *and* $F(V \setminus U) \equiv 0$.
  We have $G(U) \not\equiv 0$ and $G(V \setminus U) \equiv 0$. Fix any $u \in U$ such that $G(u) = 1$. In addition, fix any $w \in T \setminus U$ and let $v = w \oplus u$. We have $G^u(v) = G(v \oplus u) = G(w) = 0$ (since $w \in V \setminus U$ and $G(V \setminus U) \equiv 0$). We also have $G^u(\vec{0}) = G(u) = 1$ where $\vec{0}$ is the origin in $\{0,1\}^n$. By S-invariance of G^u, it follows that $G^u(S) \equiv 1$ and $G^u(T \setminus S) \equiv 0$. In particular, G^u is non-constant on T.

Since G^u is syntactically S-invariant and non-constant on T and $\mathrm{depth}(G^u) = (d-1)+1$, we may apply the induction hypothesis to G^u. Thus, we have

$$\mathrm{size}(G) = \mathrm{size}(G^u) \geq 2^{(d-1)((m/a)^{1/(d-1)}-1)}. \tag{2}$$

Since $d \geq 2$, Lemma 2 tells us

$$a + (d-1)(m/a)^{1/(d-1)} \geq d(m/a)^{1/d}. \tag{3}$$

Putting together (1), (2), (3), we get the desired bound

$$\begin{aligned}\mathrm{size}(F) &\geq 2^{a-1} \cdot 2^{(d-1)((m/a)^{1/(d-1)}-1)} \\ &= 2^{a+(d-1)(m/a)^{1/(d-1)}-d} \\ &\geq 2^{d(m^{1/d}-1)}.\end{aligned}$$

This completes the proof of Theorem 1. ◀

5 Further Remarks and Open Questions

5.1 Another Application of Theorem 1

Theorem 1 applies to interesting subspaces U besides the even-weight subspace P. Here we describe one example. Let G be a simple graph with n edges, so that $\{0,1\}^n$ is identified with the set of spanning subgraphs of G. The *cycle space* of G is the linear subspace $Z \subseteq \{0,1\}^n$ consisting of *even subgraphs* of G (i.e. spanning subgraphs in which every vertex has even degree). Consider the even-weight subspace $Z_0 = \{z \in Z : |z| \text{ is even}\}$. Provided G is non-bipartite, Z_0 is a codimension-1 subspace of Z.

Let $m = \min\{|z| : z \in Z_0^\perp \setminus Z^\perp\}$ as in Theorem 1 with $U = Z_0$ and $V = Z$. It is not hard to show that m is equal to the minimum number of edges whose removal makes G bipartite. (It follows that $m = n - c$ where c is the number edges in a maximum cut in G.) Moreover, if G is a uniform random 3-regular graph on $\frac{2}{3}n$ vertices, then $m = \Omega(n)$ asymptotically almost surely [5]. By these observations, we have:

▶ **Corollary 11.** *Let* $Z \subseteq \{0,1\}^n$ *be the cycle space of a random 3-regular graph with* n *edges, and let* $Z_0 = \{z \in Z : |z| \text{ is even}\}$. *Then a.a.s. every syntactically* Z_0*-invariant depth* $d+1$ *formula that computes* PARITY_n *over* Z *has size* $2^{d(\Omega(n)^{1/d}-1)}$.

5.2 The (U, V)-Search Problem

For linear subspaces $U \subset V$ of $\{0,1\}^n$, consider the following (U,V)-*search problem*: there is a hidden vector $x \in V \setminus U$ and the goal is to learn a nonzero coordinate of x (i.e. any $i \in [n]$ such that $w_i = 1$) by asking queries (i.e. yes/no questions) in the form of linear functions $\{0,1\}^n \to \{0,1\}$. The *d-round query complexity* of this problem is the minimum number of queries required by protocols that solve this problem on all $w \in V \setminus U$ by asking

queries in d consecutive batches (thus, 1-round = non-adaptive). By a slightly simpler version of the argument in the proof of Theorem 1, we can show a $d(m^{1/d}-1)$ lower bound on the d-round query complexity of the (U,V)-search problem for all $U \subset V$ where $m = \min\{|x| : x \in U^\perp \setminus V^\perp\}$.

This (U,V)-search problem is, in some sense, related to the Karchmer-Wigderson game where Alice gets $u \in U$ and Bob gets $v \in V \setminus U$ and their common goal is to learn a nonzero coordinate of $u \oplus v$. For an appropriate definition of "U-invariant protocols" (i.e. whatever comes from syntactically U-invariant formulas), we can translate the pair (u,v) to $(0, u \oplus v)$ without loss of generality and it becomes Alice's task to learn a nonzero coordinate of $u \oplus v$ by asking linear queries.

5.3 Open Questions

We conclude by mentioning some open questions and challenges raised by this work:

1. Does the lower bound of Theorem 1 (or something weaker like $2^{m^{\Omega(1/d)}}$) hold under the weaker assumption of semantic U-invariance, in place of syntactic U-invariance? What about Corollary 11?
2. Considering leafsize (rather than size, i.e. the number of depth-1 subformulas), improve the lower bound of Theorem 1 from $2^{d(m^{1/d}-1)}$ to $m \cdot 2^{d(m^{1/d}-1)}$.
3. Improve the upper bound of Proposition 3 from $n \cdot 2^{dn^{1/d}}$ to $O(n \cdot 2^{d(n^{1/d}-1)})$ for all $d \leq \lceil \log n \rceil$.
4. What is the maximum gap, if any, between U-invariant [depth d] formula size and non-invariant [depth d] formula size?

Acknowledgements. I thank the anonymous referees for their helpful comments.

References

1 Miklos Ajtai. Symmetric systems of linear equations modulo p. In *TR94-015 of the Electronic Colloquium on Computational Complexity*, 1994.

2 Matthew Anderson and Anuj Dawar. On symmetric circuits and fixed-point logics. *Theory of Computing Systems*, pages 1–31, 2016.

3 Andreas Blass, Yuri Gurevich, and Saharon Shelah. Choiceless polynomial time. *Ann. Pure & Applied Logic*, 100(1–3):141–187, 1999.

4 Andreas Blass, Yuri Gurevich, and Saharon Shelah. On polynomial time computation over unordered structures. *Journal of Symbolic Logic*, 67(3):1093–1125, 2002.

5 Béla Bollobás. The isoperimetric number of random regular graphs. *European Journal of combinatorics*, 9(3):241–244, 1988.

6 Anuj Dawar. On symmetric and choiceless computation. In *International Conference on Topics in Theoretical Computer Science*, pages 23–29. Springer, 2015.

7 Anuj Dawar, David Richerby, and Benjamin Rossman. Choiceless polynomial time, counting and the Cai-Fürer-Immerman graphs. *Annals of Pure and Applied Logic*, 152:31–50, 2008.

8 Larry Denenberg, Yuri Gurevich, and Saharon Shelah. Definability by constant-depth polynomial-size circuits. *Information and Control*, 70(2-3):216–240, 1986.

9 Erich Grädel and Martin Grohe. Is polynomial time choiceless? In *Fields of Logic and Computation II*, pages 193–209. Springer, 2015.

10 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC'86: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

11 Neil Immerman. *Descriptive complexity*. Springer, 2012.

12 V. M. Khrapchenko. Complexity of the realization of a linear function in the class of π-circuits. *Mathematical Notes of the Academy of Sciences of the USSR*, 9(1):21–23, 1971.

13 Martin Otto. The logic of explicitly presentation-invariant circuits. In *International Workshop on Computer Science Logic*, pages 369–384. Springer, 1996.

14 Søren Riis and Meera Sitharam. Generating hard tautologies using predicate logic and the symmetric group. *Logic Journal of IGPL*, 8(6):787–795, 2000.

15 Benjamin Rossman. Choiceless computation and symmetry. In *Fields of logic and computation*, pages 565–580. Springer, 2010.

16 Benjamin Rossman. The average sensitivity of bounded-depth formulas. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 424–430. IEEE, 2015.

On the Complexity of Quantified Integer Programming

Dmitry Chistikov[*1] **and Christoph Haase**[2]

1 **Centre for Discrete Mathematics and its Applications (DIMAP) & Department of Computer Science, University of Warwick, Warwick, UK; and Department of Computer Science, University of Oxford, Oxford, UK**
d.chistikov@warwick.ac.uk, dmitry.chistikov@cs.ox.ac.uk

2 **Department of Computer Science, University of Oxford, Oxford, UK**
christoph.haase@cs.ox.ac.uk

Abstract

Quantified integer programming is the problem of deciding assertions of the form $\mathrm{Q}_k \boldsymbol{x}_k \ldots \forall \boldsymbol{x}_2 \exists \boldsymbol{x}_1 : A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ where vectors of variables $\boldsymbol{x}_k, \ldots, \boldsymbol{x}_1$ form the vector $\boldsymbol{x}$, all variables are interpreted over $\mathbb{N}$ (alternatively, over $\mathbb{Z}$), and A and $\boldsymbol{c}$ are a matrix and vector over $\mathbb{Z}$ of appropriate sizes. We show in this paper that quantified integer programming with alternation depth k is complete for the kth level of the polynomial hierarchy.

1998 ACM Subject Classification G.2 Discrete Mathematics

Keywords and phrases integer programming, semi-linear sets, Presburger arithmetic, quantifier elimination

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.94

1 Introduction

The problem of integer programming is, given a system of linear inequalities $A \cdot \boldsymbol{x} \geq \boldsymbol{b}$, to decide whether there exists a solution for $\boldsymbol{x}$ in the non-negative integers. This problem has been studied for decades, and its 0–1 version (in which the components of $\boldsymbol{x}$ are constrained to be either 0 or 1) is one of Karp's seminal 21 NP-complete problems [8]. In this paper, we study quantified integer programming (QIP), an extension of integer programming where some of the variables can be quantified universally – so that its instances have the form

$$\mathrm{Q}_k \boldsymbol{x}_k \ \ldots \ \forall \boldsymbol{x}_2 . \ \exists \boldsymbol{x}_1 : A \cdot \boldsymbol{x} \geq \boldsymbol{c} \tag{1}$$

where $\mathrm{Q}_i \in \{\exists, \forall\}$ and $\boldsymbol{x}$ consists of all first-order variables appearing in the vectors $\boldsymbol{x}_i$.

Our main contribution is settling the complexity of QIP with k quantifier blocks (as above): we prove this problem complete for the kth level of the polynomial hierarchy, similarly to the quantified version of SAT.[1] We also show that QIP with an unbounded number of quantifier blocks is PSPACE-hard and decidable in $\mathsf{STA}(*, 2^{n^{O(1)}}, n) \subseteq \mathsf{EXPSPACE}$.[2]

* Supported by the ERC grant AVS-ISS (648701).

1 As in the case of quantified CNF SAT, the innermost block of universal quantifiers, if present, is disregarded; e.g., the $\forall^*\exists^*\forall^*$ fragment is complete for Π_2^{P}. So we find fragments of QIP complete for $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$, Π_2^{P}, Σ_3^{P}, ... , but not for $\mathsf{coNP} = \Pi_1^{\mathsf{P}}$, Σ_2^{P}, ...

2 The complexity class $\mathsf{STA}(s(n), t(n), a(n))$ was introduced by Berman [1] and contains all decision problems that can be decided by an alternating Turing machine in time $t(n)$ using space at most $s(n)$ and alternating at most $a(n)$ times on every computation branch.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 94; pp. 94:1–94:13

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

Related work and discussion. While the decidability of QIP is immediate – it can be viewed as a syntactic fragment of Presburger arithmetic, the (decidable) first-order theory of the natural numbers with addition and order, in which matrix formulas are constrained to be conjunctions of linear inequalities – its computational complexity has been unknown. It is, of course, not difficult to see that QIP (and in fact Presburger arithmetic) is PSPACE-complete if the interpretation of every first-order variable x_i is restricted to an interval $[l_i, u_i]$ that is given as part of the input: $x_i \in [l_i, u_i]$; see, e.g., [14]. But if $x_i \in \mathbb{N}$, then the best known upper bounds seem to be $\mathsf{STA}(*, 2^{2^{n^{O(1)}}}, O(n)) \subseteq$ 2-EXPSPACE, the generic upper bound for deciding Presburger arithmetic [1], and the $(k-1)$th level of the weak EXP hierarchy for the fragment with k quantifier blocks [6]. The best known lower bound has been Π_2^{P}, established recently by the authors for Π_2-instances of QIP [3, Sec. 4.2].

It may be surprising, and certainly was to the authors, that the complexity of QIP, a natural decision problem, has not yet been established. The main reason is probably that standard quantifier-elimination and automata-based techniques – which are at the core of decision procedures for Presburger arithmetic – fail to yield tight upper bounds for QIP:

- Weispfenning shows that quantifier-elimination procedures for Presburger arithmetic run in time $2^{O(|\Phi|^{(4j)^k})}$ [17, Thm. 2.1], where $|\Phi|$ denotes the size of an input formula Φ with k quantifier blocks and at most j variables in each quantifier block, and that this upper bound is essentially tight [18, Thm. 3.1]. In particular, even the NP upper bound for standard integer programming instances (Σ_1-IP) cannot be obtained by quantifier elimination.
- Automata-based decision procedures for Presburger arithmetic do not suffice either to obtain the bounds for QIP that we establish in this paper. Klaedtke shows [9, Thm. 4.6] that the size of the minimal deterministic finite automaton (DFA) for a formula Φ is upper-bounded by $2^{|\Phi|^{(j+1)^{(k+4)}}}$, which does not give any complexity bounds asymptotically better than those obtained via quantifier elimination.
- Yet another approach to QIP is to construct the semi-linear representation of the set of solutions to the system of linear inequalities of the matrix formula, and then to repeatedly project and complement this set. By an application of [2, Thm. 21], this approach gives a Π_2^{P} upper bound for the Π_2-fragment of QIP; however, as every complementation step increases the number of generators of semi-linear sets by one exponential, this approach would only yield a non-elementary upper bound for general QIP instances and fail to place fragments with bounded alternation depth inside PSPACE.

Our main results are, in short, obtained by means of a new quantifier elimination procedure on *hybrid linear sets*, which are semi-linear sets that represent sets of solutions to systems of linear inequalities. While *existential* projection ($L \mapsto \{x \mid \exists y.\ (x, y) \in L\}$) is a trivial operation on semi-linear sets (in generator representation), in this paper we define a dual operation, which we call *universal projection* ($L \mapsto \{x \mid \forall y.\ (x, y) \in L\}$), and show that its application enables us to eliminate blocks of universal quantifiers without resorting to double complementation ($\forall = \neg\exists\neg$; this would lead to a non-elementary blowup). We spell out (these and other) results of the paper in more detail in Section 3 and outline the techniques in Section 4.

Concurrently with our work and building upon a theorem of Kannan [7], Nguyen and Pak [11] have shown that Presburger arithmetic with fixed number of variables *and* fixed Boolean structure of the matrix formula (and, by necessity, where the total number of occurrences of atomic predicates is fixed) can be solved in polynomial time.

2 Preliminaries

By $\mathbb{Z}$ and $\mathbb{N}$ we denote the sets of integers and non-negative integers, respectively. Given sets X and Y, we denote by $X \Rightarrow Y$ the set of all functions with domain X and co-domain Y. Let $\mathcal{X}$ be a countably infinite set of first-order variables, and with no loss of generality assume some total ordering $\prec$ on $\mathcal{X}$. Given a finite set $X \subseteq \mathcal{X}$, an *X-indexed integer vector* is a function $\boldsymbol{v} \in (X \Rightarrow \mathbb{Z})$, and an *$X$-indexed non-negative integer vector* is a function $\boldsymbol{v} \in (X \Rightarrow \mathbb{N})$. We often call $\boldsymbol{v}$ just an *integer vector* respectively a *(non-negative) vector* when X is clear from the context. Due to the total ordering on $\mathcal{X}$, we can interchangeably write $\boldsymbol{v}$ as a tuple $(v_1, \ldots, v_n) \in \mathbb{Z}^n$ such that $n = |X|$. We denote by $\boldsymbol{e}_i$ the ith unit vector (mapping the ith variable to 1 and all other variables to 0). Addition and multiplication of a vector by a scalar value are defined component-wise. Given a set of non-negative vectors $V \subseteq \mathbb{N}^n$, its *complement* is defined as $\overline{V} := \{\boldsymbol{w} \in \mathbb{N}^n : \boldsymbol{w} \notin V\}$.

A *vector of (first-order) variables over $X \subseteq \mathcal{X}$* is a tuple $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in X^\ell$ such that each $y_i \in X$ and $y_i \prec y_{i+1}$. For $\boldsymbol{v}_i \in (X_i \Rightarrow \mathbb{Z})$, $i \in \{1, 2\}$, with $X_1 \cap X_2 = \emptyset$, by $\boldsymbol{v}_1 \circ \boldsymbol{v}_2$ we denote the vector from $(X_1 \cup X_2) \Rightarrow \mathbb{Z}$ that agrees with $\boldsymbol{v}_i$ on X_i for both $i \in \{1, 2\}$. Given a vector $\boldsymbol{v} \in (X \Rightarrow \mathbb{N})$ and a vector of variables $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in X^\ell$, the *projection of $\boldsymbol{v}$ removing variables $\boldsymbol{y}$* is the vector $\pi_{\boldsymbol{y}}(\boldsymbol{v}) \in ((X \setminus \{y_1, \ldots, y_\ell\}) \Rightarrow \mathbb{N})$ such that $\pi_{\boldsymbol{y}}(\boldsymbol{v})(x) := \boldsymbol{v}(x)$ for all $x \in X \setminus \{y_1, \ldots, y_\ell\}$. This definition of projection naturally extends to sets of vectors:

$$\pi_{\boldsymbol{y}}(V) := \bigcup_{\boldsymbol{v} \in V} \{\pi_{\boldsymbol{y}}(\boldsymbol{v})\} = \{\boldsymbol{v}_1 \mid \text{there is a } \boldsymbol{v}_2 \in (\{y_1, \ldots, y_\ell\} \Rightarrow \mathbb{N}) \text{ such that } \boldsymbol{v}_1 \circ \boldsymbol{v}_2 \in V\}.$$

For sets of vectors $V \subseteq (X \Rightarrow \mathbb{N})$, we additionally define the *universal projection*

$$\pi^*_{\boldsymbol{y}}(V) := \overline{\pi_{\boldsymbol{y}}(\overline{V})} = \{\boldsymbol{v}_1 \mid \text{for all } \boldsymbol{v}_2 \in (\{y_1, \ldots, y_\ell\} \Rightarrow \mathbb{N}) \text{ the vector } \boldsymbol{v}_1 \circ \boldsymbol{v}_2 \text{ is in } V\}.$$

For a vector $\boldsymbol{v} \in \mathbb{Z}^n$, we denote by $\|\boldsymbol{v}\| := \max\{\max_{x \in X} |\boldsymbol{v}(x)|, 2\}$ the *maximum norm* of $\boldsymbol{v}$. For $V \subseteq \mathbb{Z}^n$, we define $\|V\| := \max_{\boldsymbol{v} \in V} \|\boldsymbol{v}\|$. For a matrix A, we define $\|A\|$ to be the norm of its set of column vectors.

Quantified integer programming (QIP). Let A be an $n \times m$ integer matrix, $\boldsymbol{x} = (x_1, \ldots, x_m) \in X^m$ a vector of first-order variables for some finite $X \subseteq \mathcal{X}$, and $\boldsymbol{c} \in \mathbb{Z}^n$. We call $\mathfrak{S}\colon A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ a *system of linear inequalities*. A *solution* to $\mathfrak{S}$ is a vector $\boldsymbol{v} \in (X \Rightarrow \mathbb{Z})$ such that $A \cdot \boldsymbol{v} \geq \boldsymbol{c}$, where "$\geq$" is interpreted component-wise. We denote by $[\![\mathfrak{S}]\!] \subseteq (X \Rightarrow \mathbb{N})$ the set of *all non-negative solutions* to $\mathfrak{S}$.

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ be vectors of first-order variables over disjoint sets of variables $X_1, \ldots, X_k$, and let $\mathfrak{S}\colon A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ be a system of linear inequalities. A *formula of* QIP is given by

$$\psi = \mathrm{Q}_k \boldsymbol{x}_k.\ \mathrm{Q}_{k-1} \boldsymbol{x}_{k-1}\ \ldots\ \mathrm{Q}_1 \boldsymbol{x}_1 : A \cdot \boldsymbol{x} \geq \boldsymbol{c},$$

where $\mathfrak{S}\colon A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ is a system of linear inequalities as above, $Q_i \in \{\exists, \forall\}$, and $Q_i \neq Q_{i+1}$ for all $1 \leq i < k$, i.e., quantifiers alternate between blocks of variables. The *size* $|\psi|$ of ψ is the number of bits required to write down ψ, where we assume binary encoding of numbers, and also that $|\psi| \geq \max\{2, n + m, \log\|A\|, \log\|\boldsymbol{c}\|\}$. The set $[\![\psi]\!] \subseteq (X \setminus (X_1 \cup \cdots \cup X_k) \Rightarrow \mathbb{N})$ of *non-negative solutions* to ψ is inductively defined as follows:

- for $k = 0$, $[\![\psi]\!] := [\![\mathfrak{S}]\!]$;
- for $k > 0$ and $\psi = \exists \boldsymbol{x}_k.\psi_k$, $[\![\psi]\!] := \pi_{\boldsymbol{x}_k}[\![\psi_k]\!]$; and
- for $k > 0$ and $\psi = \forall \boldsymbol{x}_k.\psi_k$, $[\![\psi]\!] := \pi^*_{\boldsymbol{x}_k}[\![\psi_k]\!]$.

A set $M \subseteq (X \Rightarrow \mathbb{N})$ is QIP-*definable* if there is a QIP-formula ψ such that $M = [\![\psi]\!]$. Whenever $X \subseteq X_1 \cup \cdots \cup X_k$, we say that ψ is a *sentence*. In this case, ψ is *valid* if $[\![\psi]\!] = \{\top\}$ where $\top$ denotes the unique function from $\emptyset$ to $\mathbb{N}$, and *invalid* if $[\![\psi]\!] = \emptyset$. If $X \setminus (X_1 \cup \cdots \cup X_k) = Y = \{y_1, \ldots, y_m\}$, we write $\psi(y_1, \ldots, y_m)$ to indicate that ψ is *open* in Y. Given $a_1, \ldots, a_m \in \mathbb{N}$, we write $\psi[a_1/x_1, \ldots, a_m/x_m]$ to denote the instance of QIP obtained from replacing every occurrence of x_i by a_i in $\mathfrak{S}$. We say that two QIP formulas ψ and ϕ are *equivalent* if $[\![\psi]\!] = [\![\phi]\!]$; note that we may always assume with no loss of generality that ψ and ϕ are open in the same set of variables.

A *(valid) instance* of the QIP problem is a (valid) sentence ψ. We call such a ψ an instance of Σ_k-IP if $Q_k = \exists$, and an instance of Π_k-IP if $Q_k = \forall$. The *alternation depth* of ψ is the number k of quantifier blocks.

Hybrid linear and semi-linear sets. Given finite sets $B, P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\} \subseteq \mathbb{N}^m$ called *base* and *period vectors*, the *hybrid linear set generated by B and P* is the set

$$L(B, P) := \{\boldsymbol{b} + \lambda_1 \cdot \boldsymbol{p}_1 + \cdots + \lambda_n \cdot \boldsymbol{p}_n : \boldsymbol{b} \in B, \lambda_i \in \mathbb{N}, 1 \leq i \leq n\}\,.$$

The representation of $L(B, P)$ as the pair B, P (written explicitly) is called the *generator representation*. If B is singleton then $L(B, P)$ is called a *linear set*; a finite union of (hybrid) linear sets is called a *semi-linear set*. For a hybrid linear set in the generator representation $L = L(B, P)$, we denote $\|L\| := \max(\max\|B\|, \max\|P\|)$.

Hybrid linear sets represent sets of solutions to systems of linear inequalities and equalities. The following bounds on the norm in the generator representation follow from [12, Cor. 1] and [2, Prop. 4].

▶ **Proposition 1.** *Let $\mathfrak{S}\colon A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ be a system of linear inequalities such that A is an $n \times m$ integer matrix. Then $[\![\mathfrak{S}]\!] = L(B, P)$ such that $\|B\|, \|P\| \leq (m \cdot \|A\| + \|\boldsymbol{c}\| + 2)^{n+m}$.*

3 Summary

The main result of this paper is the following theorem.

▶ **Theorem 2.** *Σ_k-IP is complete for Σ_k^{P} if k is odd, and Π_k-IP is complete for Π_k^{P} if k is even.*

What happens if the parity of k is different? In this case the innermost quantifiers are universal, and it turns out that they can be eliminated in a trivial way.

▶ **Corollary 3.** *Σ_{k+1}-IP is complete for Σ_k^{P} if k is odd, and Π_{k+1}-IP is complete for Π_k^{P} if k is even.*

The lower bound of Theorem 2 is proved by a reduction from an alternating version of the subset sum problem, which is essentially shown complete for the respective levels of the polynomial-time hierarchy by Travers [15]. Our reduction and more details are given in Section 7.

The upper bound of Theorem 2 is more challenging. Note that in the well-known case of Σ_1-IP, i.e., of the standard integer programming, in order to prove membership of the problem in NP, one needs to obtain polynomial upper bounds on the bit size of minimal solutions to systems of integer linear inequalities. Such bounds were derived by, e.g., von zur Gathen and Sieveking [16]. In our work, we build upon these bounds and generalize them from Σ_1-IP instances to QIP instances.

▶ **Proposition 4** (Small Witness Property). *For a* QIP *instance* ψ *of the form* (1) *with* k *quantifier blocks, the validity of* ψ *does not change if variables of the vector* $\boldsymbol{x}_k$ *(bound by the quantifiers of the outermost block) are interpreted over* $[0, M-1]$ *instead of* $\mathbb{N}$*, where* $\log M = |\psi|^{O(k)}$.

The domains of other variables can then be bounded in turn as follows – which places QIP with fixed alternation depth into PH.

▶ **Proposition 5** (Relativization-Type Theorem). *For a* QIP *instance* ψ *of the form* (1) *with* k *quantifier blocks, the validity of* ψ *does not change if, for each* $i \in [1, k]$*, all variables of the vector* $\boldsymbol{x}_i$ *(bound by the quantifiers of the* i*th innermost block) are interpreted over* $[0, M_i - 1]$ *instead of* $\mathbb{N}$*, where* $\log M_i = |\psi|^{O(2k-i)}$ *and the constant of* $O(\cdot)$ *is independent of* ψ*,* k*, and* i.

Let us point out that in Proposition 5 it is not possible to substitute $[0, M-1]$ for the range of *all* variables; not only using $M = \max M_i$, but in fact using any finite M. For example, the sentence $\forall x.\, \exists y : y = x + 1$ is true if x and y are interpreted in $\mathbb{N}$, but false if they are interpreted in any finite segment $[0, M-1]$.

▶ Remark. The last observation, of course, also holds for Presburger arithmetic in general: any relativization-type theorem (analogous to Proposition 5) must assign different ranges to variables from different quantifier blocks; for instance, this reveals a flaw in the formulation of the relativization-type Theorem 2.2 in [17].

Notice that our small witness property (Proposition 4) is specific to QIP, in the sense that its bound is smaller by one exponential compared to its analogue for general Presburger formulas [17, Thm. 2.2] (the latter is, in fact, tight, as shown implicitly in, e.g., [5, 6]). At the core of our small witness property is a new quantifier elimination procedure for QIP:

▶ **Proposition 6** (Quantifier Elimination). *Given a* QIP *formula* $\phi(\boldsymbol{x})$ *with alternation depth* k*, there exists an equivalent* Σ_1-IP *formula* $\phi'(\boldsymbol{x})$ *with at most* $2^{|\psi|^{O(k)}}$ *existentially quantified variables and numbers of absolute value bounded by* $2^{|\psi|^{O(k)}}$.

The ideas behind Propositions 4 and 6 are outlined in the following Section 4.

Further results. Our results give a uniform upper bound for the general QIP problem, where the number of quantifier blocks can be unbounded. For such a QIP instance, our relativization-type theorem (Proposition 5) suggests doubly exponential ranges for all variables, which places QIP in the complexity class $\mathsf{STA}(*, 2^{n^{O(1)}}, n)$, as $k \le n$. The best lower bound is PSPACE, by the arguments of Section 7.

Another by-product of our techniques is a pseudo-polynomial algorithm for QIP in which the total number of variables is fixed and the matrix formula is $A \cdot \boldsymbol{x} = \boldsymbol{c}$ instead of $A \cdot \boldsymbol{x} \ge \boldsymbol{c}$.

In terms of auxiliary techniques, on the way to our quantifier elimination procedure for QIP we discover (in Sections 5 and 6) some new properties of hybrid linear sets. In particular, these properties enable us to find, as a side result, a polynomial-time algorithm for universality of hybrid linear sets in the generator representation, even if all input numbers are written in binary (Proposition 17 in Section 5).

Finally, our results extend in a natural way to the version of quantified integer programming where all variables are interpreted over $\mathbb{Z}$ instead of over $\mathbb{N}$: the results of Theorem 2 and Corollary 3 still hold.

4 Main ideas

As explained in Section 3, bounding the range of the outermost quantifier is the main technical task in our development. In this section we explain how to do this, thus sketching the ideas behind both the small witness property (Proposition 4) and the quantifier elimination procedure (Proposition 6).

Suppose we start with a QIP instance ψ of the form (1); to find a suitable upper bound M_k for the range of the $\boldsymbol{x}_k$ variables of ψ, we will compute generator representations for the sets of models of formulas

$$\psi_j(\boldsymbol{x}_k, \ldots, \boldsymbol{x}_{j+1}) = \mathrm{Q}_j \boldsymbol{x}_j \ \ldots \ \forall \boldsymbol{x}_2.\ \exists \boldsymbol{x}_1 : A \cdot \boldsymbol{x} \geq \boldsymbol{c}$$

for all $j \in [0, k]$, where, as previously, $\boldsymbol{x}$ is the concatenation of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$. For each value of the parameter j, we will find upper bounds on the integers appearing in these representations, starting with $j = 0$ and culminating with $j = k$. The upper bound for the value of parameter $j = k$ will be a valid choice for M_k.

Let us now describe this computation in more detail. Consider a simple abstract example, a Σ_3-IP instance with 3 variables, $\psi\colon \exists x.\ \forall y.\ \exists z : A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ where $\boldsymbol{x} = (x, y, z)$. Let $L_0 \subseteq \mathbb{N}^3$ be the set of all models of $\psi_0\colon A \cdot \boldsymbol{x} \geq \boldsymbol{c}$; this is a hybrid linear set – denote it $L(C_0, Q_0)$ – with $\|C_0\|$, $\|Q_0\|$ upper-bounded by a polynomial in $\|A\|, \|\boldsymbol{c}\|$ with degree at most the size of ψ (see, e.g., Proposition 1). It follows that $\log\|C_0\|$ and $\log\|Q_0\|$ are polynomial in the size of ψ. It is clear that the the set $L_1 = [\![\psi_1]\!] = \{(x, y) \in \mathbb{N}^2 \mid \text{there exists a } z \in \mathbb{N} \text{ such that } (x, y, z) \in L_0\}$ is simply a projection of $L(C_0, Q_0)$, and in particular $L_1 = L(C, Q)$ where the sets C and Q are obtained by removing z-coordinates from all vectors in C_0 and Q_0, respectively. Hence, $\log\|C\|$ and $\log\|Q\|$ are also polynomial in the size of ψ. (This will, of course, work for all occurrences of the existential quantifier in ψ, including $\exists x$ in our example; but we will need to handle the universal quantifier $\forall y$ before handling $\exists x$.)

The next step is to transform the generator representation $L(C, Q)$ of the set $L_1 = [\![\psi_1]\!]$ into a generator representation of the set

$$L_2 = [\![\psi_2]\!] = \{x \in \mathbb{N} \mid \text{for all } y \in \mathbb{N} \text{ it holds that } (x, y) \in L_1\}.$$

This set L_2 is the *universal projection* of $L(C, Q)$: $L_2 = \pi_y^*(L(C, Q))$; cf. Section 2. As the main technical contribution of the present paper, we show that, in general, (*i*) universal projections of hybrid linear sets are hybrid linear sets themselves and that (*ii*) universal projection as an operation on hybrid linear sets can only lead to a moderate increase in the magnitude of generators. (These results are summarized in Proposition 10 below. For the usual projection, such facts are obvious.)

We now briefly introduce the techniques that we develop for handling the universal projection. Define for each $y \in \mathbb{N}$ the cross section $S(y) = \{x \in \mathbb{N} \mid (x, y) \in L_1\}$, then

$$L_2 = \bigcap_{y \in \mathbb{N}} S(y) \tag{2}$$

by definition. Each set $S(y)$ is a semi-linear set (and, in fact, a hybrid linear set – because it is essentially the intersection of two hybrid linear sets, see Lemma 13, and such intersections are hybrid linear sets themselves, see, e.g., [2, Theorem 6]), but the intersection in (2) is infinite, and, in general, an infinite intersection of semi-linear sets does not have to be semi-linear.[3] However, we prove (in Section 5) the following lemma, which is our first and key insight:

[3] For every $n \geq 1$, consider the hybrid linear set $L_n = \mathbb{N} \setminus \{0, n\} = L([1, n-1] \cup \{2n\}, \{n\})$. Given any $A \subseteq \mathbb{N}$, the intersection $\bigcap_{n \in A} L_n = \mathbb{N} \setminus (\{0\} \cup A)$ is only semi-linear (i.e., ultimately periodic) if so is A.

▶ **Lemma 7.** *Let $L = L(C,Q) \subseteq \mathbb{N}^m$ be a hybrid linear set with $C, Q \subseteq \mathbb{N}^m$. Let the components of vectors be indexed by m variables X, let $U \subseteq X$, $|U| = s$, and suppose $\boldsymbol{u}$ is the corresponding vector of variables. Then the following statements hold:*

- *If, for some variable $u_i \in U$, the set Q contains no multiple of the unit vector $\boldsymbol{e}_i$ associated with u_i, then $\pi^*_{\boldsymbol{u}}(L) = \emptyset$.*
- *Otherwise, denote $a_i = \min\{a \mid a \cdot \boldsymbol{e}_i \in Q\}$ and $H = \{\boldsymbol{b} \in \mathbb{N}^s \mid 0 \le \boldsymbol{b}(u_i) \le a_i - 1 \text{ for all } u_i \in U\}$. Then*

$$\pi^*_{\boldsymbol{u}}(L) = \bigcap_{\boldsymbol{b} \in H} \pi_{\boldsymbol{u}}\big(L(C,Q) \cap \{\boldsymbol{u} = \boldsymbol{b}\}\big)$$

where $\{\boldsymbol{u} = \boldsymbol{b}\}$ denotes the hybrid linear set $\{\boldsymbol{c} \in \mathbb{N}^m \mid \boldsymbol{c}(u_i) = \boldsymbol{b}(u_i) \text{ for all } u_i \in U\}$.

In other words, unless $L_2 = \emptyset$, the intersection in (2) can be made finite without changing its result: $\bigcap_{y \in \mathbb{N}} S(y) = \bigcap_{y < N} S(y)$, where $\log N$ is polynomial in the size of ψ. Since, as we have just mentioned, hybrid linear sets are closed under finite intersections, this shows that the set L_2 is hybrid linear, and, in fact, the following general result follows:

▶ **Proposition 8.** *A set in $\mathbb{N}^m$ is* QIP*-definable iff it is hybrid linear.*

Furthermore, the set L_2 turns out to have a small generator representation as well. Indeed, we first observe that all sets $S(y)$ have representations $L(B_y, P)$ with a common set of periods P and with $\|B_y\|$, $\|P\|$ small if so is $\|y\|$ (Lemma 13 in Section 5). We then prove (in Section 6) the following lemma, which is our second insight:

▶ **Lemma 9.** *Let $L_i = L(C_i, Q)$, $i \in [1,n]$, be hybrid linear sets with $C_i, Q \subseteq \mathbb{N}^m$. The set $S = \bigcap_{i=1}^n L_i$ has a representation $S = L(B,Q)$ where $\|B\| \le \max_{i \in [1,n]} \|L_i\|^{O(m^3)}$ independently of n.*

In other words, long intersections of hybrid linear sets with a common set of periods preserve small representations, regardless of the number of sets in the intersection. Combining Lemmas 7 and 9, we obtain the following statement:

▶ **Proposition 10.** *Let $L = L(C,Q) \subseteq \mathbb{N}^m$ be a hybrid linear set with $C, Q \subseteq \mathbb{N}^m$. Let the components of vectors be indexed by m variables $\boldsymbol{u}, \boldsymbol{v}$, and suppose the vector $\boldsymbol{u}$ has s variables. Then the universal projection $\pi^*_{\boldsymbol{u}}(L)$ has a representation $L(B,P)$ where $P = \pi_{\boldsymbol{u}}(\{\boldsymbol{q} \in Q \mid q_1 = \ldots = q_s = 0\})$ and $\|B\| \le \|L\|^{O(m^5)}$.*

In particular, we conclude that the set $L_2 = \bigcap_{y < N} L(B_y, P)$ has a representation $L(B,P)$ with $\|B\| < M$ where $\log M$ is polynomial in the size of ψ. But note that $\psi = \psi_3$ is true iff $L_2 = [\![\psi_2]\!]$ is non-empty; therefore, the validity of ψ is unchanged if the range of $\exists x$ is changed from $\mathbb{N}$ to $[0, M-1]$. Thus, in our example the bound M_3 can be chosen as M; it can hence be deduced that a Σ_3^{P} algorithm can handle such instances. The argument for the general case follows the same lines.

5 Universal projection and universality

A semi-linear set in $\mathbb{N}^d$ is called *universal* if it is equal to $\mathbb{N}^d$.

▶ **Example 11.** A one-dimensional hybrid linear set $L = L(B,P) \subseteq \mathbb{N}$ with $B, P \subseteq \mathbb{N}$ is universal iff $P \setminus \{0\} \neq \emptyset$ and L contains the integer segment $[0, k-1]$ where $k = \min P \setminus \{0\}$. Indeed, the right-to-left direction is immediate: if L satisfies the conditions above, then $\mathbb{N} = L([0,k-1], \{k\}) \subseteq L$. For the left-to-right direction, suppose $L = \mathbb{N}$. First observe that the set $P \setminus \{0\}$ is non-empty because L is infinite. Therefore, $k > 0$ is well-defined. Second, as $L = \mathbb{N}$, the set L contains all natural numbers, in particular those in $[0, k-1]$.

The following lemma generalizes Example 11; recall that $\boldsymbol{e}_i$ denotes the ith unit vector.

▶ **Lemma 12.** *A hybrid linear set $L = L(B, P) \subseteq \mathbb{N}^m$ with $B, P \subseteq \mathbb{N}^m$ is universal iff P contains vectors $a_i \cdot \boldsymbol{e}_i$ for some $a_i > 0$, for every $i \in [1, m]$, and L contains the box $H = [0, a_1 - 1] \times \ldots \times [0, a_m - 1]$.*

Proof. The right-to-left direction is immediate: if L satisfies the conditions of the lemma, then $\mathbb{N}^m = L(H, \{a_1 \cdot \boldsymbol{e}_1, \ldots, a_m \cdot \boldsymbol{e}_m\}) \subseteq L$. For the left-to-right direction, suppose $L = \mathbb{N}^m$. We first prove that, for each $i \in [1, m]$, the set of periods P contains a vector $a_i \cdot \boldsymbol{e}_i$ with $a_i > 0$. Assume without loss of generality that $i = 1$ and denote $N = \mathbb{N} \times \mathbf{0} \subseteq \mathbb{N}^m$. Since L is universal (and $\mathbb{Q}_{\geq 0} \times \mathbf{0}$ is a face of $\mathbb{Q}^m_{\geq 0}$), $N = L(B, P) \cap N \subseteq L(B \cap N, P \cap N)$. Therefore, the set $P \cap N$ contains at least one vector $a_1 \cdot \boldsymbol{e}_1$ with $a_1 > 0$, otherwise the set N would be finite. Hence, P contains $a_1 \cdot \boldsymbol{e}_1, \ldots, a_m \cdot \boldsymbol{e}_m$ with all $a_i > 0$. It now remains to note that, as $L = \mathbb{N}^m$, the set L contains all nonnegative integer vectors, in particular those in H. This completes the proof. ◀

▶ Remark. If $m = 1$, then in the statement of Lemma 12, the condition $H \subseteq L(B, P)$ is equivalent to the condition $H \subseteq B$, as long as H is defined using the *shortest* vector $a_1 \cdot \boldsymbol{e}_1$ in $P \setminus \{\mathbf{0}\}$. For $m \geq 2$, this is no longer the case.

▶ **Lemma 13.** *Suppose $L = L(C, Q) \subseteq \mathbb{N}^m$ and $M = L(D, E) \subseteq \mathbb{N}^m$ where $E = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_s\}$. Then the set $L \cap M$ has a representation $L(B, P)$ where $P = \{\boldsymbol{q} \in Q \mid q_{s+1} = \ldots = q_m = 0\}$ and $\|B\| \leq \|L\|^{O(m^2)} \cdot \|M\|^{O(m)}$.*

We can now restate and prove Lemma 7, which appeared previously in Section 4.

▶ **Lemma 14.** *Let $L = L(C, Q) \subseteq \mathbb{N}^m$ be a hybrid linear set with $C, Q \subseteq \mathbb{N}^m$. Let the components of vectors be indexed by m variables X, let $U \subseteq X$, $|U| = s$, and suppose $\boldsymbol{u}$ is the corresponding vector of variables. Then the following statements hold:*

- *If, for some variable $u_i \in U$, the set Q contains no multiple of the unit vector $\boldsymbol{e}_i$ associated with u_i, then $\pi^*_{\boldsymbol{u}}(L) = \emptyset$.*
- *Otherwise, denote $a_i = \min\{a \mid a \cdot \boldsymbol{e}_i \in Q\}$ and $H = \{\boldsymbol{b} \in \mathbb{N}^s \mid 0 \leq \boldsymbol{b}(u_i) \leq a_i - 1$ for all $u_i \in U\}$. Then*

$$\pi^*_{\boldsymbol{u}}(L) = \bigcap_{\boldsymbol{b} \in H} \pi_{\boldsymbol{u}}\big(L(C, Q) \cap \{\boldsymbol{u} = \boldsymbol{b}\}\big)$$

where $\{\boldsymbol{u} = \boldsymbol{b}\}$ denotes the hybrid linear set $\{\boldsymbol{c} \in \mathbb{N}^m \mid \boldsymbol{c}(u_i) = \boldsymbol{b}(u_i)$ for all $u_i \in U\}$.

Proof. Denote $V = X \setminus U$; we will abuse notation and let symbols $\boldsymbol{u}$ and $\boldsymbol{v}$ refer to U- and V-indexed integer vectors (wherever this creates no confusion). By definition, a vector $\boldsymbol{v}^*$ belongs to $\pi^*_{\boldsymbol{u}}(L)$ if and only if for all $\boldsymbol{u}$ the vector $(\boldsymbol{u}, \boldsymbol{v}^*)$ belongs to L. This condition is equivalent to the requirement that

$$L \cap \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{N}^m \mid \boldsymbol{v} = \boldsymbol{v}^*\} = \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{N}^m \mid \boldsymbol{v} = \boldsymbol{v}^*\}. \tag{3}$$

Note that $\{(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{N}^m \mid \boldsymbol{v} = \boldsymbol{v}^*\} = L((\mathbf{0}, \boldsymbol{v}^*), E)$ where E is the set of all unit vectors associated with variables $\boldsymbol{u}$. We now apply Lemma 13: $L \cap L((\mathbf{0}, \boldsymbol{v}^*), E) = L(D_{\boldsymbol{v}^*}, R)$ where $R = \{\boldsymbol{q} = (\boldsymbol{u}, \boldsymbol{v}) \in Q \mid \boldsymbol{v} = \mathbf{0}\}$. Now the requirement (3) has the form $L(D_{\boldsymbol{v}^*}, R) = \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{N}^m \mid \boldsymbol{v} = \boldsymbol{v}^*\}$ and, by Lemma 12, is equivalent to the requirement that, first, the set R contains some multiple of the unit vector, $a_i \cdot \boldsymbol{e}_i$ for some $a_i > 0$, associated with each variable $u_i \in U$, and, second, the set $L(D_{\boldsymbol{v}^*}, R)$ contains the box

$$H(\boldsymbol{v}^*) = \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{N}^m \mid \boldsymbol{v} = \boldsymbol{v}^*, 0 \leq u_i \leq a_i - 1 \text{ for all variables } u_i \in U\}.$$

Note that in the statement of Lemma 12 we can always choose $a_i \cdot \boldsymbol{e}_i$ to be the shortest vectors of the required form in R; expanding the definition of R then gives

$$a_i = \min\{a \mid a \cdot \boldsymbol{e}_i \in Q\} \qquad \text{for each variable } u_i \in U. \tag{4}$$

We now make the following observations. First, the set R does not depend on the vector $\boldsymbol{v}^*$, but only on Q and on the way the variables are split into $\boldsymbol{u}$ and $\boldsymbol{v}$. Therefore, the condition that R contains $a_i \cdot \boldsymbol{e}_i$ for some $a_i > 0$ is either satisfied or not satisfied for all $\boldsymbol{v}^*$ simultaneously. In the former case, $\pi^*_{\boldsymbol{u}}(L) = \emptyset$; so it suffices to consider the latter case. We have the following equivalence:

$$\boldsymbol{v}^* \in \pi^*_{\boldsymbol{u}}(L) \quad \text{iff} \quad (\boldsymbol{u}, \boldsymbol{v}^*) \in L(D_{\boldsymbol{v}^*}, R) \text{ for all } \boldsymbol{u} \in H$$

where $H = \{\boldsymbol{u} \mid 0 \le u_i \le a_i - 1 \text{ for all variables } u_i \in U\}$ and a_i are as defined in (4). Since $L(D_{\boldsymbol{v}^*}, R)$ was chosen as $L \cap \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{N}^m \mid \boldsymbol{v} = \boldsymbol{v}^*\}$, this is the same as

$$\boldsymbol{v}^* \in \pi^*_{\boldsymbol{u}}(L) \quad \text{iff} \quad (\boldsymbol{u}, \boldsymbol{v}^*) \in L \text{ for all } \boldsymbol{u} \in H,$$

and the equation of the lemma follows. ◀

▶ **Example 15.** Consider any set $L = L(C, \{3\,\boldsymbol{e}_2\}) \subseteq \mathbb{N}^2$ with a finite $C \subseteq \mathbb{N}^2$. Its universal projection $L' = \pi^*_y(L) = \{x \mid (x, y) \in L \text{ for all } y \in \mathbb{N}\}$ can be obtained by taking cross sections $S_b = \{x \mid (x, b) \in L\}$ for $b = 0, 1, 2$, removing the y coordinate, and intersecting the results: $L' = \pi_y(S_0) \cap \pi_y(S_1) \cap \pi_y(S_2)$ where the projection $\pi_y \colon \mathbb{N}^2 \to \mathbb{N}$ removes the y coordinate. So whether or not a specific $a \in \mathbb{N}$ belongs to L' is fully determined by whether the vectors $(a, 0)$, $(a, 1)$, and $(a, 2)$ belong to L. In fact, this conclusion will also hold if instead of L we consider any set $M = L(C, \{3\,\boldsymbol{e}_2\} \cup Q)$ where Q contains no vectors of the form $a \cdot \boldsymbol{e}_2$.

Intermezzo: Deciding universality of hybrid linear sets

The technique developed above, in fact, enables us to show that universality of hybrid linear sets (given in generator representation) can be decided in polynomial time, even if all numbers are written in binary. Consider the following lemma, which is a more general version of Example 11 and Lemma 12.

▶ **Lemma 16.** *Let $L = L(B, P) \subseteq \mathbb{N}^m$ be a hybrid linear set with $B, P \subseteq \mathbb{N}^m$. Define the set of* shallow points*,*

$$W = \big\{\, \boldsymbol{w} \in \mathbb{N}^m \mid \textit{there is no } \boldsymbol{p} \in P \setminus \{\boldsymbol{0}\} \textit{ with } \boldsymbol{w} \ge \boldsymbol{p} \,\big\} = \mathbb{N}^m \setminus \bigcup_{\boldsymbol{p} \in P \setminus \{\boldsymbol{0}\}} (\boldsymbol{p} + \mathbb{N}^m).$$

Then L is universal iff $W \subseteq B$.

Indeed, for Example 11, observe that for $m = 1$ the set W is the integer segment $[0, k-1]$ where $k = \min P \setminus \{0\}$; cf. Remark 5.

For Lemma 12, note that $W \subseteq B$ is only possible if W is finite, which implies that for each $i \in [1, m]$ there is a vector $a_i \cdot \boldsymbol{e}_i \in P$ with $a_i > 0$ (otherwise all such vectors for some given i are in W, and there are infinitely many of them). But then $W \subseteq H = [0, a_1 - 1] \times \ldots \times [0, a_m - 1]$.

▶ **Proposition 17.** *There is a polynomial-time algorithm that takes a hybrid linear set $L(B, P) \subseteq \mathbb{N}^m$, presented as $B, P \subseteq \mathbb{N}^m$ with numbers written in binary, and decides if $L(B, P)$ is universal.*

Proof. By the characterization of Lemma 16, it is sufficient to check if $W \subseteq B$. First check that the necessary condition of Lemma 12 is satisfied: if for some i there is no $a_i \cdot \boldsymbol{e}_i \in P$ with $a_i > 0$, then $L(B, P)$ is not universal. Otherwise consider the Hasse diagram of the partial order $(H, \leq)$, i.e., the directed acyclic graph with vertex set H and all edges $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} < \boldsymbol{y}$ and there is no $\boldsymbol{z}$ with $\boldsymbol{x} < \boldsymbol{z} < \boldsymbol{y}$. Notice that this graph does not have to be of polynomial size with respect to the input.

Run the depth-first search (DFS) procedure on (a part of) this graph, starting from $\mathbf{0}$ and for each $\boldsymbol{x} \in H$ ordering the outgoing edges $(\boldsymbol{x}, \boldsymbol{y})$ according to the (unique) index $i \in [1, m]$ for which $x_i < y_i$. Whenever the current node is outside W, the algorithm backtracks (observe that the set W is always downward closed, i.e., whenever $\boldsymbol{w} \in W$ and $\boldsymbol{w}' \leq \boldsymbol{w}$, then also $\boldsymbol{w}' \in W$); if it is in W but not in B, the algorithm terminates immediately, reporting that $L(B, P)$ is not universal. If the search finishes, the algorithm concludes that $W \subseteq B$ and reports that $L(B, P)$ is universal. All visited nodes are marked and not re-entered, ensuring that no node is ever visited twice. As all visited notes are checked for inclusion in B, which is given as part of the input, it follows that the running time of the search is proportional to the size of the input, and the entire procedure works in polynomial time. ◀

6 Long intersections

▶ **Lemma 18.** *Let $L_i = L(C_i, Q)$, $i \in [1, n]$, be hybrid linear sets with $C_i, Q \subseteq \mathbb{N}^m$. Suppose the vectors of Q are linearly independent. Then the set $S = \bigcap_{i=1}^{n} L_i$ has a representation $S = L(B, Q)$ where $\|B\| \leq 2^{O(m \log m)} \cdot \max_{i \in [1,n]} \|L_i\| \cdot \|Q\|^m$ independently of n.*

Proof (Sketch). Note that $\bigcap_{i=1}^{n} L(C_i, Q)$ is the union over all $\boldsymbol{c}_1 \in C_1, \ldots, \boldsymbol{c}_n \in C_n$ of $\bigcap_{i=1}^{n} L(\boldsymbol{c}_i, Q)$, so we shall assume with no loss of generality that $C_i = \{\boldsymbol{c}_i\}$ for all i.

Define a point lattice $\mathcal{L} = Q \cdot \mathbb{Z}^r = \{Q \cdot \boldsymbol{u} \mid \boldsymbol{u} \in \mathbb{Z}^r\}$ where $r = |Q|$; see, e.g., [10, Chapter 2]. Vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^m$ are called *congruent* modulo $\mathcal{L}$, written $\boldsymbol{x} \equiv \boldsymbol{y} \pmod{\mathcal{L}}$, if and only if $\boldsymbol{x} - \boldsymbol{y} \in \mathcal{L}$. This congruence splits $\mathbb{Z}^m$ into a disjoint union of equivalence classes, which have the form $\boldsymbol{x} + \mathcal{L}$ where $\boldsymbol{x} \in \mathbb{Z}^m$. Note that the relation $\equiv$ is compatible with addition and subtraction of elements of Q, in the sense that vectors $\boldsymbol{x} \pm \boldsymbol{q}$, $q \in Q$, belong to the same equivalence class as $\boldsymbol{x}$; therefore, $L_i = \boldsymbol{c}_i + Q \cdot \mathbb{N}^r \equiv \boldsymbol{c}_i \pmod{\mathcal{L}}$. Hence, unless the intersection $\bigcap_{i=1}^{n} L_i$ is empty, it must be the case that $\boldsymbol{c}_i \equiv \boldsymbol{c}_j \pmod{\mathcal{L}}$ for all i, j. We assume in the sequel that this is indeed the case, i.e., all sets L_i are contained in the same equivalence class $\boldsymbol{c}_1 + \mathcal{L}$.

Let us now define the coordinates in $\boldsymbol{c}_1 + \mathcal{L}$ in a natural way. Consider the mapping $\psi \colon \boldsymbol{c}_1 + \mathcal{L} \to \mathbb{Z}^r$ that maps each $\boldsymbol{x}$ into a vector $\boldsymbol{u} = \psi(\boldsymbol{x})$ such that $\boldsymbol{x} = \boldsymbol{c}_1 + Q \cdot \boldsymbol{u}$; note that $\boldsymbol{u}$ exists as long as $\boldsymbol{x} \in \boldsymbol{c}_1 + \mathcal{L}$ and is determined uniquely because the vectors in Q are linearly independent. The mapping ψ is, in fact, a bijection between $\boldsymbol{c}_1 + \mathcal{L}$ and $\mathbb{Z}^r$, so $L_1 \cap \ldots \cap L_n = \psi^{-1}(\psi(L_1) \cap \ldots \psi(L_n))$. Denote $\boldsymbol{f}_i = \psi(\boldsymbol{c}_i)$ and observe that $\psi(L_i) = \psi(\boldsymbol{c}_i) + \mathbb{N}^r$. So a vector $\boldsymbol{v} \in \mathbb{Z}^r$ belongs to the intersection of all $\psi(L_i)$ if and only if $\boldsymbol{v} \geq \boldsymbol{f}_i$ for all $i \in [1, n]$. This condition is satisfied if and only if $\boldsymbol{v} \geq \boldsymbol{f}$ where $\boldsymbol{f}$ is the component-wise maximum of vectors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$; in other words, $\bigcap_{i=1}^{n} \psi(L_i) = \boldsymbol{f} + \mathbb{N}^r$ and $L_1 \cap \ldots \cap L_n = L(\psi^{-1}(\boldsymbol{f}), Q)$.

It remains to find an upper bound on $\|\psi^{-1}(\boldsymbol{f})\|$. Note that $\psi^{-1}(\boldsymbol{f}) = \boldsymbol{c}_1 + Q \cdot \boldsymbol{f}$, so $\|\psi^{-1}(\boldsymbol{f})\| \leq \|\boldsymbol{c}_1\| + \|Q \cdot \boldsymbol{f}\|$. Suppose $\boldsymbol{f} = (f^1, \ldots, f^r)$ and $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_r\}$, then $Q \cdot \boldsymbol{f} = f^1 \cdot \boldsymbol{q}_1 + \ldots + f^r \cdot \boldsymbol{q}_r$. Recall that each f^j is, in fact, a component of some $\boldsymbol{f}_i = \psi(\boldsymbol{c}_i)$.

For this $i = i(j)$ it holds that $\boldsymbol{c}_i = \boldsymbol{c}_1 + Q \cdot \boldsymbol{f}_i$, and by [2, Proposition 3] we have

$$|f^j| \leq 2^{O(m \log m)} \cdot \max\left(\|\boldsymbol{c}_i - \boldsymbol{c}_1\|, \|Q\|\right) \cdot \|Q\|^{m-1} \quad \text{and}$$
$$\|\psi^{-1}(\boldsymbol{f})\| \leq \|C_1\| + m \cdot \max_{j \in [1,r]} \|f^j \cdot \boldsymbol{q}_j\| \leq 2^{O(m \log m)} \cdot \max_{i \in [1,n]} \|L_i\| \cdot \|Q\|^m. \quad \blacktriangleleft$$

We can now restate and prove Lemma 9, which appeared previously in Section 4.

▶ **Lemma 19.** *Let $L_i = L(C_i, Q)$, $i \in [1,n]$, be hybrid linear sets with $C_i, Q \subseteq \mathbb{N}^m$. The set $S = \bigcap_{i=1}^n L_i$ has a representation $S = L(B, Q)$ where $\|B\| \leq \max_{i \in [1,n]} \|L_i\|^{O(m^3)}$ independently of n.*

Proof (Sketch). We first apply a discrete version of the Carathéodory theorem [2, Proposition 5] to the set L_1, decomposing it into a union of hybrid linear sets with linearly independent periods:

$$L_1 = \bigcup_j M_j \quad \text{where} \quad M_j = L(D_j, Q_j) \quad \text{and} \quad \|D_j\| \leq \|C_1\| + (\#Q \cdot \|Q\|)^{O(m)},$$

with each $Q_j \subseteq Q$ a set of linear independent vectors (here and below $\#$ denotes the cardinality of a set). The intent is to make it possible to invoke Lemma 18.

Notice that, whereas intersecting two hybrid linear sets L and L' with sets of periods P and $P' \subseteq P$, respectively, will always give a hybrid linear set with the set of periods P' (see, e.g., [2, Theorem 6] and, transitively, Theorem 5.6.1 of [4, p. 180]), this observation would not suffice for our purposes. Indeed, the magnitude of the base vectors in the hybrid linear representation of $L \cap L'$ can still increase compared to the magnitude of the base vectors of L and L'; and so $n-1$ consecutive applications of this operation would lead to a blowup in the representation size if n grows. Instead of using this observation, we will rely on Lemma 18 to defeat the effect of large n, and will use another trick to make its application possible.

Indeed, observe that

$$L_1 \cap L_2 \cap \ldots \cap L_n = \bigcup_j M_j \cap L_2 \cap \ldots \cap L_n = \bigcup_j (M_j \cap L_2) \cap \ldots \cap (M_j \cap L_n).$$

Since the sets of periods of M_j and L_i are Q_j and Q, respectively, it follows by [2, Theorem 6] that each $M_j \cap L_i$ is a hybrid linear set with representation $L(B_{i,j}, Q_j)$, where

$$\|B_{i,j}\| \leq ((\#Q_j + \#Q) \cdot \max(\|M_j\|, \|L_i\|))^{O(m)} \leq \max\left(\|C_1\| + (\#Q \cdot \|Q\|)^{O(m)}, \|L_i\|\right)^{O(m)}.$$

But now, for each j, the intersection of $L(B_{i,j}, Q_j)$, $i \in [2,n]$, satisfies the conditions of Lemma 18, and thus can be written as $L(B_j, Q_j)$ with $\|B_j\|$ small with respect to $\|B_{i,j}\|$ and $\|Q_j\|$ (estimations to follow). Now $S = \bigcup_j L(B_j, Q_j)$, and it remains to note that, as $L_i + L(\mathbf{0}, Q) = L_i$ for all i, it also holds that $S + L(\mathbf{0}, Q) = S$ and hence

$$S = \bigcup_j L(B_j, Q_j) + L(\mathbf{0}, Q) = \bigcup_j L(B_j, Q) = L\left(\bigcup_j B_j,\ Q\right), \quad \text{with}$$
$$\|B_j\| \leq 2^{O(m \log m)} \cdot \max\left(\max_{i \in [2,n]} \|B_{i,j}\|, \|Q_j\|\right) \cdot \|Q_j\|^m \leq \max_{i \in [1,n]} \|L_i\|^{O(m^3)}. \quad \blacktriangleleft$$

7 Lower bounds

We show lower bounds for QIP and Σ_k-IP via a reduction from a generalisation of the classical SubsetSum problem. For odd k, let $\boldsymbol{a}_k \in \mathbb{N}^{m_k}, \ldots, \boldsymbol{a}_1 \in \mathbb{N}^{m_1}$ be vectors of natural

numbers, and let $t \in \mathbb{N}$ be a target. An instance of Σ_k-SubsetSum is a tuple $(\boldsymbol{a}_k, \ldots, \boldsymbol{a}_1, t)$. This instance is a valid instance whenever the following holds:

$$\exists \boldsymbol{x}_k \in \{0,1\}^{m_k}. \, \forall \boldsymbol{x}_{k-1} \in \{0,1\}^{m_{k-1}} \, \ldots \, \exists \boldsymbol{x}_1 \in \{0,1\}^{m_1} : \sum_{i=1}^{k} \boldsymbol{a}_i \cdot \boldsymbol{x}_i = t. \tag{5}$$

Thus, Σ_k-SubsetSum can be viewed as the 0–1 variant of Σ_k-IP, i.e., variables are only interpreted over $\{0,1\}$. For even k, Π_k-SubsetSum is defined analogously. When we take the union of Σ_k-SubsetSum for all $k > 0$, we obtain QSubsetSum.

▶ **Proposition 20.** *For every fixed $k > 0$, for odd k Σ_k-SubsetSum is Σ_k^{P}-complete, and for even k Π_k-SubsetSum is Π_k^{P}-complete.* QSubsetSum *is* PSPACE*-complete.*

Upper bounds for Σ_k-SubsetSum and QSubsetSum can be obtained trivially. The PSPACE lower bound for QSubsetSum was established by Travers in [15, Lem. 4]. Unfortunately, the construction given in [15] does not directly yield Σ_k^{P} hardness for Σ_k-SubsetSum, as the lower bound for QSubsetSum is shown in [15] by a reduction from 3-CNF QBF in which the alternating quantifiers range over *single* variables, and Σ_k^{P} hardness for 3-CNF k-QBF requires an unbounded number of variables in every quantifier block [13]. It is not difficult to show that the construction from [15] can indeed be adapted in order to yield Σ_k^{P} hardness for Σ_k-SubsetSum for odd k, and likewise for even k.

Proof of lower bounds in Theorem 2

We reduce from Σ_k-SubsetSum and show how to transform an instance given as (5) into an equivalent instance of Σ_k-IP. Note that the existentially quantified variables do not present an issue, since, for instance, $\boldsymbol{x}_1 \in \{0,1\}^{m_1}$ iff $\boldsymbol{x}_1 \leq \mathbf{1}$, i.e., (5) is equivalent to

$$\exists \boldsymbol{x}_k \in \{0,1\}^{m_k}. \, \forall \boldsymbol{x}_{k-1} \in \{0,1\}^{m_{k-1}} \, \ldots \, \forall \boldsymbol{x}_2 \in \{0,1\}^{m_2}. \, \exists \boldsymbol{x}_1 : \sum_{i=1}^{k} \boldsymbol{a}_i \cdot \boldsymbol{x}_i = t \wedge \boldsymbol{x}_1 \leq \mathbf{1}. \tag{6}$$

The key insight is that, for universally quantified variables, conjunctions of linear integer constraints can express division with remainder using any fixed divisor. In particular, consider

$$\exists \boldsymbol{x}_k \in \{0,1\}^{m_k}. \, \forall \boldsymbol{x}_{k-1} \in \{0,1\}^{m_{k-1}} \, \ldots \, \forall \boldsymbol{x}_2. \, \exists \boldsymbol{x}_1. \, \exists \boldsymbol{\lambda} : \\ \sum_{i=3}^{k} \boldsymbol{a}_i \cdot \boldsymbol{x}_i + \boldsymbol{a}_2 \cdot (\boldsymbol{x}_2 - 2 \cdot \boldsymbol{\lambda}) + \boldsymbol{a}_1 \cdot \boldsymbol{x}_1 = t \wedge \boldsymbol{x}_1 \leq \mathbf{1} \wedge \mathbf{0} \leq \boldsymbol{x}_2 - 2 \cdot \boldsymbol{\lambda} \leq \mathbf{1}. \tag{7}$$

We claim that the sentences (6) and (7) are equivalent. First, no matter what $\boldsymbol{x}_2$ is, $\boldsymbol{\lambda}$ has to be $\lfloor \boldsymbol{x}_2/2 \rfloor$ in order to satisfy the last constraint of (7). If sentence (6) is true, then (7) is also true. Indeed, if $\boldsymbol{x}_2 \in \{0,1\}^{m_2}$, then we can choose $\boldsymbol{\lambda} = \mathbf{0}$ and the inequalities become the same as before (and thus, for instance, there is an appropriate $\boldsymbol{x}_1$). Analogously, if $\boldsymbol{x}_2$ is outside $\{0,1\}^m$, then it is the vector $\boldsymbol{x}_2 - 2 \cdot \boldsymbol{\lambda}$ that is in $\{0,1\}^{m_2}$, and for this vector we already know the appropriate $\boldsymbol{x}_1$ from the previous formula. Conversely, suppose the sentence (7) is true, then it is in particular true for all choices $\boldsymbol{x}_2 \in \{0,1\}^{m_2}$ in which case $\boldsymbol{\lambda} = \lfloor \boldsymbol{x}_2/2 \rfloor = \mathbf{0}$. Hence, the assignment for $\boldsymbol{x}_1$ chosen in (7) given $\boldsymbol{x}_2$ will also work for (6). This proves the claim.

In fact, the trick above works regardless of how many universal variables we have and at which positions they occur in the quantifier prefix. So we can handle both existential and universal variables and can transform any instance of Σ_k-SubsetSum respectively Π_k-SubsetSum into an equivalent instance of Σ_k-IP respectively Π_k-IP, which yields the desired lower bounds, when variables are interpreted over the natural numbers.

References

1 Leonard Berman. The complexitiy of logical theories. *Theor. Comput. Sci.*, 11:71–77, 1980. doi:10.1016/0304-3975(80)90037-7.

2 Dmitry Chistikov and Christoph Haase. The taming of the semi-linear set. In *Automata, Languages, and Programming, ICALP*, volume 55 of *LIPIcs*, pages 128:1–128:13. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.128.

3 Dmitry Chistikov, Christoph Haase, and Simon Halfon. Context-free commutative grammars with integer counters and resets. *Theor. Comput. Sci.*, pages –, 2017. To appear. doi:10.1016/j.tcs.2016.06.017.

4 Seymour Ginsburg. *The mathematical theory of context-free languages*. McGraw-Hill, 1966.

5 Erich Grädel. Dominoes and the complexity of subclasses of logical theories. *Ann. Pure Appl. Logic*, 43(1):1–30, 1989. doi:10.1016/0168-0072(89)90023-7.

6 Christoph Haase. Subclasses of Presburger arithmetic and the weak EXP hierarchy. In *Computer Science Logic and Logic in Computer Science, CSL-LICS*, pages 47:1–47:10. ACM, 2014. doi:10.1145/2603088.2603092.

7 Ravi Kannan. Test sets for integer programs, ∀∃ sentences. In *Polyhedral Combinatorics, Proceedings of a DIMACS Workshop, Morristown, New Jersey, USA, June 12-16, 1989*, pages 39–48, 1990.

8 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

9 Felix Klaedtke. Bounds on the automata size for Presburger arithmetic. *ACM Trans. Comput. Log.*, 9(2):11:1–11:34, 2008. doi:10.1145/1342991.1342995.

10 Jiri Matoušek. *Lectures on discrete geometry*. Graduate texts in mathematics. Springer, 2002. doi:10.1007/978-1-4613-0039-7.

11 Danny Nguyen and Igor Pak. Complexity of short Presburger arithmetic. In *Symposium on the Theory of Computing, STOC*, 2017. To appear. URL: https://arxiv.org/abs/1704.00249.

12 Loïc Pottier. Minimal solutions of linear Diophantine systems: Bounds and algorithms. In *Rewriting Techniques and Applications, RTA*, volume 488 of *Lect. Notes Comp. Sci.*, pages 162–173. Springer, 1991. doi:10.1007/3-540-53904-2_94.

13 Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976. doi:10.1016/0304-3975(76)90061-X.

14 K. Subramani. Tractable fragments of Presburger arithmetic. *Theory Comput. Syst.*, 38(5):647–668, 2005. doi:10.1007/s00224-004-1220-0.

15 Stephen D. Travers. The complexity of membership problems for circuits over sets of integers. *Theor. Comput. Sci.*, 369(1-3):211–229, 2006. doi:10.1016/j.tcs.2006.08.017.

16 Joachim von zur Gathen and Malte Sieveking. A bound on solutions of linear integer equalities and inequalities. *P. Am. Math. Soc.*, 72(1):155–158, 1978. doi:10.1090/S0002-9939-1978-0500555-0.

17 Volker Weispfenning. The complexity of almost linear Diophantine problems. *J. Symb. Comp.*, 10(5):395–403, 1990. doi:10.1016/S0747-7171(08)80051-X.

18 Volker Weispfenning. Complexity and uniformity of elimination in Presburger arithmetic. In *Symbolic and Algebraic Computation, ISSAC*, pages 48–53. ACM, 1997. doi:10.1145/258726.258746.

Word Equations in Nondeterministic Linear Space*

Artur Jeż

Institute of Computer Science, University of Wrocław, Wrocław, Poland
aje@cs.uni.wroc.pl

Abstract

Satisfiability of word equations is an important problem in the intersection of formal languages and algebra: Given two sequences consisting of letters and variables we are to decide whether there is a substitution for the variables that turns this equation into true equality of strings. The computational complexity of this problem remains unknown, with the best lower and upper bounds being, respectively, NP and PSPACE. Recently, the novel technique of recompression was applied to this problem, simplifying the known proofs and lowering the space complexity to (nondeterministic) $\mathcal{O}(n \log n)$. In this paper we show that satisfiability of word equations is in nondeterministic linear space, thus the language of satisfiable word equations is context-sensitive. We use the known recompression-based algorithm and additionally employ Huffman coding for letters. The proof, however, uses analysis of how the fragments of the equation depend on each other as well as a new strategy for nondeterministic choices of the algorithm, which uses several new ideas to limit the space occupied by the letters.

1998 ACM Subject Classification F.4.3 [Formal Languages] Decision Problems, Classes Defined by Grammars or Automata, F.4.2 Grammars and Other Rewriting Systems, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Word equations, string unification, context-sensitive languages, space efficient computations, linear space

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.95

1 Introduction

Solving *word equations* was an intriguing problem since the dawn of computer science, motivated first by its ties to Hilbert's 10th problem. Initially it was conjectured that this problem is undecidable, which was disproved in a seminal work of Makanin [10]. At first little attention was given to computational complexity of Makanin's algorithm and the problem itself; these questions were reinvestigated in the '90 [6, 18, 9], culminating in the EXPSPACE implementation of Makanin's algorithm by Gutiérrez [5].

The connection to compression was first observed by Plandowski [16], who showed that a length-minimal solution of size N has a compressed representation of size $\mathsf{poly}(n, \log N)$. Plandowski further explored this approach [14] and proposed a PSPACE algorithm [13], which is the best bound up to date; a simpler PSPACE solution also based on compression was proposed by Jeż [8]. On the other hand, this problem is only known to be NP-hard, and it is conjectured that it is in NP.

The importance of these mentioned algorithms lays with the possibility to extend them (in nontrivial ways) to various scenarios: free groups [11, 1, 3], representation of all solutions [15, 8, 17], traces [12, 2], graph groups [4], terms [7] and others.

* Work supported under National Science Centre, Poland project number 2014/15/B/ST6/00615.

While the computational complexity of word equations remains unknown, its exact space complexity is intriguing as well: Makanin's algorithm uses exponential space [5], Plandowski [13] gave no explicit bound on the space usage of his algorithm, a rough estimation is $\mathsf{NSPACE}(n^5)$, the recent solution of Jeż [8] yields $\mathsf{NSPACE}(n \log n)$. Moreover, for $\mathcal{O}(1)$ variables a linear bound on space complexity was shown [8]; recall that languages recognisable in nondeterministic linear space are exactly the context-sensitive languages.

In this paper we show that satisfiability of word equations can be tested in nondeterministic linear space in terms of the number of bits of the input, thus showing that the language of satisfiable word equations is context-sensitive (and by the famous Immerman–Szelepcsényi theorem: the language of unsatisfiable word equations). The employed algorithm is a (variant of) algorithm of Jeż [8], which additionally uses Huffman coding for letters in the equation. On the other hand, the actual proof uses a different encoding of letters, which extends the ideas used in a (much simpler) proof in case of $\mathcal{O}(1)$ variables [8, Section 5]; the other new ingredient is a different strategy of compression: roughly speaking, previously a strategy that minimised the length of the equation was used. Here, a more refined strategy is used: it simultaneously minimises the size of a particular bit encoding, enforces that changes in the equation (during the algorithm) are local, and limits the amount of new letters that are introduced to the equation.

The bound holds when letters and variables in the input are encoded using an arbitrary encoding, in particular, the Huffman coding (so the most efficient one) is allowed.

2 The (known) algorithm

We first present a slight variation of the algorithm of Jeż [8] and the notions necessary to understand how it works. The proofs are omitted, yet they should be intuitively clear.

Notions. The word equation is a pair (U, V), written as $U = V$, where $U, V \in (\Gamma \cup \mathcal{X})^*$ and Γ and $\mathcal{X}$ are disjoint alphabets of *letters* and *variables*, both are collectively called *symbols*. By n_X we denote the number of occurrences of X in the (current) equation; in the algorithm n_X does not change till X is removed from the equation, in which case n_X becomes 0. A *substitution* is a morphism $S : \mathcal{X} \cup \Gamma \to \Gamma'^*$, where $\Gamma' \supseteq \Gamma$ and $S(a) = a$ for every $a \in \Gamma$, a substitution naturally extends to $(\mathcal{X} \cup \Gamma)^*$. A *solution* of an equation $U = V$ is a substitution S such that $S(U) = S(V)$; given a solution S of an equations $U = V$ we call $S(U)$ the *solution word*. We allow the solution to use letters that are not present in the equation, this does not change the satisfiability: all such letters can be changed to a fixed letter from Γ, and the obtained substitution is still a solution. Yet, the proofs become easier, when we allow the usage of such letters. The alphabet Γ' is usually given implicitly: as the set of letters used by the substitution. A *block* is a string a^ℓ with $\ell \geq 1$ that cannot be extended to the left nor to the right with a.

As we deal with linear-space, the encoding used by the input equation matters. We assume only that the input is given by a fixed (uniquely decodable) coding: each symbol in the input is always given by the same bitstring and given the bitstrings representing the sides of the equation there is only one pair of strings (over $\Gamma \cup \mathcal{X}$) that is encoded in this way. It is folklore that among such codes the Huffman code yields the smallest space consumption (counted in bits) and moreover the Huffman coding can be efficiently computed, also in linear space. As we focus on space counted in bits and use encodings, by $||\alpha||$ we denote the space consumption of the encoding of α, the encoding shall be always clear from the context. Furthermore, whenever we talk about space complexity, it is counted in bits.

Nondeterministic Linear Space. We recall some basic facts about the nondeterministic space-bounded computation. A nondeterministic procedure *is sound*, when given a unsatisfiable word equation $U = V$ it cannot transform it to a satisfiable one, regardless of the nondeterministic choices; a procedure is *complete*, if given a satisfiable equation $U = V$ for some nondeterministic choices it returns a satisfiable equation $U' = V'$. A composition of sound (complete) procedures is sound (complete, respectively). It is enough that we show linear-space bound for one particular computation: as the bound is known, we limit the space available to the algorithm and reject the computations exceeding it.

The algorithm. We use (a variant of) recompression algorithm [8], which conceptually applies the following two operations on $S(U)$ and $S(V)$: given a string w and alphabet Γ

- the Γ block compression of w is a string w' obtained by replacing every block a^ℓ, where $a \in \Gamma$ and $\ell \geq 2$, with a fresh letter a_ℓ;
- the (Γ_ℓ, Γ_r) pair compression of w, where Γ_ℓ, Γ_r is a partition of Γ, is a string w' obtained by replacing every occurrence of a pair $ab \in \Gamma_\ell\Gamma_r$ with a fresh letter c_{ab}.

A *fresh* letter means that it is not currently used in the equation, nor in Γ, yet each occurrence of a fixed ab is replaced with the same letter. The a_ℓ and c_{ab} are just notation conventions, the actual letters in w' do not store the information how they were obtained. For shortness, we call Γ block compression the Γ compression or block compression, when Γ is clear from the context; similar convention applies to (Γ_ℓ, Γ_r) pair compression, called (Γ_ℓ, Γ_r) compression or pair compression, when (Γ_ℓ, Γ_r) is clear from the context. We say that a pair $ab \in \Gamma_\ell\Gamma_r$ is *covered* by a partition Γ_ℓ, Γ_r.

The intuition is that the algorithm aims at performing those compression operations on the solution word and to this end it modifies the equation a bit and then performs the compression operations on U and V (and conceptually also on the solution, i.e. on $S(X)$ for each variable X). Below we describe, how it is performed on the equation.

BlockComp: For the equation $U = V$ and the alphabet Γ of letters in this equation for each variable X we first guess the first and last letter of $S(X)$ as well as the lengths ℓ, r of the longest prefix consisting only of a, called a-prefix, and b-suffix (defined similarly) of $S(X)$. Then we replace X with $a^\ell X b^r$ (or $a^\ell b^r$ or a^ℓ when $S(X) = a^\ell b^r$ or $S(X) = a^\ell$); this operation is called *popping a-prefix and b-suffix*. Then we perform the Γ-block compression on the equation (this is well defined, as we can treat variables as symbols from outside Γ).

PairComp: For the alphabet Γ, which will always be the alphabet of letters in the equation right before the block compression we partition Γ into Γ_ℓ and Γ_r (in a way described in Section 3.2) and then for each variable X guess whether $S(X)$ begins with a letter $b \in \Gamma_r$ and if so, replace X with bX or b, when $S(X) = b$, and then do a symmetric action for the last letter and Γ_ℓ; this operation is later referred to as *popping letters*. Then we perform the (Γ_ℓ, Γ_r) compression on the equation.

LinWordEq works in *phases*, until an equation with both sides of length 1 is obtained: in a single phase it establishes the alphabet Γ of letters in the equation, performs the Γ compression and then repeats: guess the partition of Γ to Γ_ℓ and Γ_r and perform the (Γ_ℓ, Γ_r) compression, until each pair $ab \in \Gamma^2$ was covered by some partition.

Correctness. Given a solution S we say that some nondeterministic choices *correspond to* S, if they are done as if LinWordEq knew S. For instance, it guesses correctly the first letter of $S(X)$ or whether $S(X) = \epsilon$. (The choice of a partition does not fall under this category.)

▶ **Lemma 1** ([8, Lemma 2.8 and Lemma 2.10])**.** BlockComp *is sound and complete; to be more precise, for any solution S of an equation $U = V$ for the nondeterministic choices*

corresponding to S the returned equation $U' = V'$ has a solution S' such that $S'(U')$ is the Γ compression of $S(U)$ and $S'(X)$ is obtained from $S(X)$ by removing the a-prefix and b-suffix, where a is the first letter of $S(X)$ and b the last, and then performing the Γ compression.

When Γ_ℓ and Γ_r are disjoint, the PairComp(Γ_ℓ, Γ_r) *is sound and complete; to be more precise, for any solution S of an equation $U = V$ for the nondeterministic choices corresponding to S the returned equation $U' = V'$ has a solution S' such that $S'(U')$ is the (Γ_ℓ, Γ_r) compression of $S(U)$ and $S'(X)$ is obtained from $S(X)$ by removing the first letter of $S(X)$, if it is in Γ_r, and the last, if it is in Γ_ℓ, and then performing the (Γ_ℓ, Γ_r) compression.*

The solution S' from Lemma 1 is called a *solution corresponding to S* after (Γ_ℓ, Γ_r) compression (Γ compression, respectively); we also talk about a *solution corresponding to S*, when the compression operation is clear from the context and extend this notion to a solution corresponding to S after a phase. What is important later on is how S' is obtained from S: it is modified as if the subprocedures knew first/last letter of $S(X)$ and popped appropriate letters from the variables and then compressed pairs/blocks in substitution for variables.

Lemma 1 yields the soundness and completeness of LinWordEq, for the termination we observe that iterating the compression operations shortens the string by a constant fraction, thus the length of a solution word shortens by a constant fraction in each phase.

▶ **Lemma 2.** *Let w be a string over an alphabet Γ and w' a string obtained from w by a Γ compression followed by a sequence of (Γ_ℓ, Γ_r) compressions (where Γ_ℓ, Γ_r is a partition of Γ) such that each pair $ab \in \Gamma^2$ is covered by some partition. Then $|w'| \leq \frac{2|w|+1}{3}$.*

▶ **Theorem 3.** LinWordEq *is sound, complete and terminates (for appropriate nondeterministic choices) for satisfiable equations. It runs in linear (bit) space.*

In the following, we will also need one more technical property of block compression.

▶ **Lemma 4.** *Consider a solution S during a phase with nondeterministic choices corresponding to S and the corresponding solution S' of $U' = V'$ after the block compression. Then $S'(U')$ has no two consecutive letters $aa \in \Gamma$.*

This is true after block compression and afterwards no letters from Γ are introduced.

Compressing blocks in small space. Storing, even in a concise way, the lengths of popped prefixes and suffixes in Γ compression makes attaining the linear space difficult. This was already observed [8] and a linear-space implementation of BlockComp was given [8]. It performs a different set of operations, yet the effect is the same as for BlockComp. Instead of explicitly naming the lengths of blocks, we treat them as integer parameters; then we declare, which maximal blocks are of the same length (those lengths depend linearly on the parameters); verifying the validity of such a guess is done by writing a system of (linear) Diophantine equations that formalise those equalities and checking its satisfiability. This procedure is described in detail in [8, Section 4]. In the end, it can be implemented in linear bitspace.

▶ **Lemma 5** ([8, Lemma 4.7])**.** BlockComp *can be implemented in space linear in the bit-size of the equation*

Huffman coding. At each step of the algorithm we encode letters (though not variables) in the equation using Huffman coding. This may mean that when going from $U = V$ to $U' = V'$ the encoding of letters changes and in fact using the former encoding in the latter equation

may lead to super-linear space (imagine that we pop from each variable a letter that has a very long code). Using standard methods changing the encoding during a transition from $U = V$ to $U' = V'$ can be done in $\mathcal{O}(||U = V|| + ||U' = V'||)$ bit-space.

▶ **Lemma 6.** *Given a string (encoded using some uniquely decodable code), its Huffman coding can be computed in linear bitspace.*

Each subprocedure of LinWordEq *that transforms an equation $U = V$ to $U' = V'$ can be implemented in bit-space $\mathcal{O}(||U = V||_1 + ||U' = V'||_2)$, where $||\cdot||_1$ and $||\cdot||_2$ are the Huffman codings for letters in $U = V$ and $U' = V'$, respectively.*

3 Space consumption

In order to bound the space consumption, we will use bit-encoding of letters that depends on the current equation. We use the term 'encoding' even though it may assign different codes to different occurrences of the same letter, but two different letters never have the same code. Since we are interested in linear space only, we do not care about the multiplicative $\mathcal{O}(1)$ factors in the space consumption and can assume that our code is prefix-free, say by terminating each encoding with a special symbol \$. We show that such an encoding uses linear space, which also shows that the Huffman encoding of the letters in the equation uses linear space, as Huffman encoding uses the smallest space among the prefix codes.

The idea of our 'encoding' is: for each letter in the current equation we establish an interval I of indices in the original equation (viewed as as string $U_0V_0[1\,..\,|U_0V_0|]$) on which it 'depends' (this has to be formalised) and encode this letter as $U_0V_0[I]\#i$, when it is ith in the sequence of letters assigned I and $U_0V_0[I]$ is the original equation restricted to indices in I The dependency is formalised in Section 3.1, while Section 3.2 first gives the high-level intuition and then upper-bound on the used space.

For technical reasons we insert into the equation ending markers at the beginning and end of U and V, i.e. write them as $@U@, @V@$ for some special symbol @. Those markers are ignored by the algorithm, yet they are needed for the encoding.

3.1 Dependency intervals

First, we need some notation. The input equation is denoted by $U_0 = V_0$, the $U = V$ and $U' = V'$ are used for the current equation and equation after performing some operation. We treat the input equation as a single string U_0V_0 and consider its *indices*, i.e. numbers from 1 to $|U_0V_0|$, denoted by letters i, i', j and intervals of such indices, denoted by letter I, I' or $[i\,..\,j]$. The $U_0V_0[I]$ and $U_0V_0[i\,..\,j]$ denotes the substring of U_0V_0 restricted to indices in $[I]$ or in $[i\,..\,j]$. We use a partial order $\leq$ on intervals: $[i\,..\,j] \leq [i'\,..\,j']$ if $i \leq i'$ and $j \leq j'$.

In the current equation, i.e. the one stored by LinWordEq, we do not consider indices but rather *positions* and denote them by letters p, q. We do not think of them as numbers but rather as pointers: when $U = V$ is transformed by some operation to $U' = V'$ but the letter/variable at position p was not affected by this transformation, we still say that this letter/variable is at position p. On the other hand, the affected letters are on positions that were not present in $U = V$. In the same spirit we denote by p the positions in $U = V$ and the corresponding position in $S(U) = S(V)$. We still use the left-to-right ordering on positions, use $p - 1$ and $p + 1$ to denote the previous and next position; we also consider intervals of positions, yet they are used rarely so that they are not confused with intervals of indices, on which we focus mostly. Given an equation $U = V$ and an interval of positions P by $UV[P]$

we denote the string of letters and variables at positions in P, again, this notation is used rarely. In the input equation the index and position is the same.

With each position p in the (current) equations (including the endmarkers) we associate *dependency interval* $\mathrm{dep}(p)$, called *depint*; if the depint is a single index $\{i\}$, we denote it i. The idea is that the letter at position p is uniquely determined by $U_0V_0[\mathrm{dep}(I)]$ (and the nondeterministic choices of the algorithm), note that it may include both variables and letters. We use the notions of $\subseteq$ and $\supseteq$ for the depints with a usual meaning; we take unions of the them, denoted by $\cup$, but only when the result is an interval. We say that I and I' are similar, denoted as $I \sim I$, if $U_0V_0[I] = U_0V_0[I']$. Given an interval I of indices in U_0V_0 by $\mathrm{Pos}(I)$ we denote positions in the current equation whose depint is I, i.e. $\mathrm{Pos}(I) = \{p \mid \mathrm{dep}(p) = I\}$. In the analysis it is also convenient to look at positions whose depint is a superset of I: $\mathrm{Pos}_{\supseteq}(I) = \{p \mid \mathrm{dep}(p) \supseteq I\}$, this is usually used for $I = \{i\}$

We shall ensure the following properties:

(I1) Given a depint I, the $\mathrm{Pos}(I)$ is an interval of positions, similarly $\mathrm{Pos}_{\supseteq}(I)$.

(I2) Given depints I, I' such that $\mathrm{Pos}(I) \neq \emptyset \neq \mathrm{Pos}(I')$ either $I \leq I'$ or $I \geq I'$.

(I3) For depints $I \sim I'$ it holds that $U_0V_0[I] = U_0V_0[I']$.

Assigning depints to letters. When X at position p pops a letter into position p' then $\mathrm{dep}(p') \leftarrow \mathrm{dep}(p)$ (which is the position of this occurrence of X in the input equation). Whenever we perform the (Γ_ℓ, Γ_r) compression then in parallel for each position p such that $UV[p] \in \Gamma_\ell$ we assign $\mathrm{dep}(p) \leftarrow \mathrm{dep}(p) \cup \mathrm{dep}(p+1)$ ($p+1$ may be a a position variable or an endmarker). Then we perform a symmetric action for positions whose letters are in Γ_r (so for $p-1$).

For Γ compression, we perform in parallel the following operation for each block (perhaps of length 1) of a letter in Γ: given a maximal block a^ℓ at positions $p, p+1, \ldots, p+\ell-1$ we set the depints of those positions to $\bigcup_{i=-1}^{\ell} \mathrm{dep}(p+i)$ (note that $p-1$ and $p+\ell$ are included).

In the following we mostly focus on $\mathrm{Pos}_{\supseteq}(i)$. As this is an interval of positions, we visualize that $\mathrm{Pos}_{\supseteq}(I)$ extends to the neighbouring positions. Thus we will refer to operations of changing the depints before the block compression and pair compression as *extending of* $\mathrm{Pos}_{\supseteq}(I)$ to new positions; those positions *get their depints extended*. Note that this notion does not apply to the case when we pop letters from variables.

Depints defined in this way satisfy the conditions (I1–I3).

▶ **Lemma 7.** *(I1–I3) hold during* LinWordEq.

Proof. We first show (I1) for $\mathrm{Pos}_{\supseteq}(i)$. The proof is by induction; this is true at the beginning. If we make a union of depints, a position adjacent to a position in $\mathrm{Pos}_{\supseteq}(i)$ symbol can become part of $\mathrm{Pos}_{\supseteq}(i)$ (this can be iterated when the depints are changed before the blocks compression), which is fine. During the compression, we compress symbols on positions with the same depints, so this is fine. When we pop a letter from variable at position p to position p' then $\mathrm{dep}(p') = \mathrm{dep}(p) \in \mathrm{Pos}_{\supseteq}(i)$ and by inductive assumption $\mathrm{Pos}_{\supseteq}(i)$ was an interval, which shows the claim.

We now show by induction that $i \leq i'$ implies $\mathrm{Pos}_{\supseteq}(i) \leq \mathrm{Pos}_{\supseteq}(i')$. Clearly this holds at the beginning, as then $\mathrm{Pos}_{\supseteq}(i) = \mathrm{Pos}(i) = \{i\}$ and $\mathrm{Pos}_{\supseteq}(i') = \mathrm{Pos}(i') = \{i'\}$. Consider the moment, in which the condition $\mathrm{Pos}_{\supseteq}(i) \leq \mathrm{Pos}_{\supseteq}(i')$ is first violated, by symmetry it is enough to consider the case in which the first position in $\mathrm{Pos}_{\supseteq}(i')$ is smaller than the first in $\mathrm{Pos}_{\supseteq}(i)$. If this position was just popped then it cannot be popped to the right, as the position of popping variable is in $\mathrm{Pos}_{\supseteq}(i')$. So it was popped to the left. But then the variable that popped it was on position in $\mathrm{Pos}_{\supseteq}(i')$ and by induction assumption $\mathrm{Pos}_{\supseteq}(i') \geq \mathrm{Pos}_{\supseteq}(i)$,

so it had a position from $\mathrm{Pos}_{\supseteq}(i)$ to its left, contradiction. The other option is that this happened when a depint of a position was changed so that it got into $\mathrm{Pos}_{\supseteq}(i')$. But then the position to its right was in $\mathrm{Pos}_{\supseteq}(i')$ and by induction assumption either this position was in $\mathrm{Pos}_{\supseteq}(i)$ or some position to the left of it was; in both cases the position also got into $\mathrm{Pos}_{\supseteq}(i)$.

Now (I1) for $\mathrm{Pos}_{\supseteq}([i\,..\,j])$ for an arbitrary depint $[i\,..\,j]$ follows: $\mathrm{Pos}_{\supseteq}([i\,..\,j]) = \bigcap_{k=i}^{j} \mathrm{Pos}_{\supseteq}(k)$ and as each $\mathrm{Pos}_{\supseteq}(k)$ is an interval, also $\mathrm{Pos}_{\supseteq}([i\,..\,j])$ is.

For the purpose of the proof, define $\mathrm{Pos}_{\subseteq}(I) = \{p \mid \mathrm{dep}(p) \subseteq I\}$ (a dual notion to $\mathrm{Pos}_{\supseteq}(I)$).

▶ **Claim 8.** *$\mathrm{Pos}_{\subseteq}(I)$ consists of consecutive positions. Given two similar depints $I \sim I'$ $UV[\mathrm{Pos}_{\subseteq}(I)]$ and $UV[\mathrm{Pos}_{\subseteq}(I')]$ are equal as sequences of symbols and the corresponding positions in them have similar depints.*

The proof follows by a simple, yet tedious induction and it is omitted.

Claim 8 is stronger than (I3) and so it implies it. Concerning (I1): $\mathrm{Pos}(I) = \mathrm{Pos}_{\supseteq}(I) \cap \mathrm{Pos}_{\subseteq}(I)$; as both are intervals, also $\mathrm{Pos}(I)$ is.

Concerning (I2), we show a stronger statement: given positions $p, p+1$ it holds that $\mathrm{dep}(p) \leq \mathrm{dep}(p{+}1)$. Let i, i' be the leftmost indices in $\mathrm{dep}(p), \mathrm{dep}(p{+}1)$, respectively. Assume for the sake of contradiction that $i > i'$. We already showed that then $\mathrm{Pos}_{\supseteq}(i) \geq \mathrm{Pos}_{\supseteq}(i')$. So if $p+1 \in \mathrm{Pos}_{\supseteq}(i') \leq \mathrm{Pos}_{\supseteq}(i) \ni p$ then also $p \in \mathrm{Pos}_{\supseteq}(i')$, i.e. $i' \in \mathrm{dep}(p)$. As $i' < i$ then the leftmost index in $\mathrm{dep}(p')$ cannot be i. The proof for rightmost index is similar. ◀

Encoding of letters. Letters in $\mathrm{Pos}(I)$ are encoded as $U_0V_0[I]\#1$, $U_0V_0[I]\#2$, etc. Note, that there is no a priori bound on the size of such numbers. Furthermore, if $I' \sim I$ then encoding $I\#i$ and $I'\#i$ is the same (these are the same symbols by (I3).

3.2 Pair compression strategy

We assume that LinWordEq makes the nondeterministic choices according to the solution, thus the space consumption of a run depends only on the choices of the partitions during pair compression, called *a strategy*. We describe a linear-space strategy.

Idea. Imagine we ensured that during one phase each variable popped $\mathcal{O}(1)$ letters and each $\mathrm{Pos}_{\supseteq}(i)$ expanded by $\mathcal{O}(1)$ letters. Then $|\mathrm{Pos}_{\supseteq}(i)| = \mathcal{O}(1)$: we introduced $\mathcal{O}(1)$ positions to $\mathrm{Pos}(i)$, say at most k, and by Lemma 2 among positions in $\mathrm{Pos}_{\supseteq}(i)$ at the beginning of the phase there were at least $2/3$ took part in compression, so their number dropped by $1/3$; thus $|\mathrm{Pos}_{\supseteq}(i)| \leq 3k$. As a result, $|\mathrm{Pos}(I)| \leq 3k$ for each depint I: as $\mathrm{Pos}(I) \subseteq \mathrm{Pos}_{\supseteq}(i)$ for $i \in I$. This would yield that the whole bit-space used for the encoding is linear: each number m used in $U_0V_0[I]\#m$ is at most $3k = \mathcal{O}(1)$, so they increase the size by at most a constant fraction. On the other hand, the depints consume:

$$\sum_{I:\mathrm{depint}} ||U_0V_0[I]|| \cdot |\mathrm{Pos}(I)| = \sum_{i:\mathrm{index}} ||U_0V_0[i]|| \cdot |\mathrm{Pos}_{\supseteq}(i)|$$

(a simple proof is given later) and the right hand side is linear in terms of the input equation: $|\mathrm{Pos}_{\supseteq}(i)| = \mathcal{O}(1)$ and $\sum_{i:\mathrm{index}} ||U_0V_0[i]||$ is the the bit-size of the input equation.

It remains to ensure that $\mathrm{Pos}_{\supseteq}(i)$ do not extend too much and variables do not pop too much letters. Given a phase, we call a letter *new*, if it was introduced during this phase. New letters cannot be popped nor can $\mathrm{Pos}_{\supseteq}(i)$ be extended to positions with new letters. Thus they are used to prevent extending $\mathrm{Pos}_{\supseteq}(i)$ and popping: it is enough to ensure that

the first/last letter of a variable is new and that a letter on the position to the left/right of $\mathrm{Pos}_{\supseteq}(I)$ is new.

Unfortunately, we cannot ensure this for all variables $\mathrm{Pos}_{\supseteq}(i)$. We can make this true *in expectation*: given a random partition there is a $1/4$ probability that a fixed pair is compressed (and the resulting letter is new). This requires formalisation and calculations.

Strategy. Given a solution S of an equation we say that a variable X is *left blocked* if $S(X)$ has at most one letter or the first or second letter in $S(X)$ is new, otherwise a variable is *left unblocked*; define *right blocked* and *right unblocked* variables similarly. An index i is *left blocked* if in $S(U)$ (or $S(V)$, respectively) there is at most position to the left of $\mathrm{Pos}_{\supseteq}(i)$ or one of the letters on the positions one and two to the left of $\mathrm{Pos}_{\supseteq}(i)$ is new, otherwise i is *left unblocked*; define *right blocked* and *right unblocked* indices similarly.

▶ **Lemma 9.** *Consider a solution $S = S_0$ and consecutive solutions $S_1, S_2, \ldots$ corresponding to it during a phase. If a variable X becomes left (right) blocked for some S_k, then it is left (right, respectively) blocked for each S_ℓ for $\ell \geq k$ and it pops to the left (right, respectively) at most 1 letter after it became left (right, respectively) blocked. If an index i becomes left (right) blocked for some S_k then it is left (right, respectively) blocked for each S_ℓ for $\ell \geq k$ and at most one letter to the left (right, respectively) will have its depint extended by i after i became left (right, respectively) blocked.*

The proof follows by a simple case inspection and it is omitted.

The strategy iterates steps 1, 2, 3 and 4. In a step i it chooses a partition so that the corresponding i-th sum below decreases by $1/2$, unless this sum is already 0:

$$\sum_{X\in\mathcal{X} \text{ left unblocked}} n_X \cdot ||X|| + \sum_{X\in\mathcal{X} \text{ right unblocked}} n_X \cdot ||X|| \tag{1}$$

$$\sum_{i\text{: left unblocked index}} ||U_0V_0[i]|| + \sum_{i\text{: right unblocked index}} ||U_0V_0[i]|| \tag{2}$$

$$\sum_{X\in\mathcal{X} \text{ left unblocked}} n_X + \sum_{X\in\mathcal{X} \text{ right unblocked}} n_X \tag{3}$$

$$\sum_{i\text{: left unblocked index}} 1 + \sum_{i\text{: right unblocked index}} 1 \tag{4}$$

The idea of the steps is: (1) upper-bounds the increase of bit-size of depints in the equation after popping letters. So by iteratively halving it we ensure that total encoding increase caused by popping letters is small. Similarly, (2) upper-bounds the increase due to expansion of indices to new depints. The following (3) is connected (in a more complex way) to an increase, after popping, of number of bits used for numbers in the encoding. Similarly (4) to an increase after the extension of depints.

▶ **Lemma 10.** *During the pair compression* LinWordEq *can always choose a partition that at least halves the value of a chosen non-zero sum among* (1)–(4).

Proof. Consider (1) and take a random partition, in the sense that each letter $a \in \Gamma$ goes to the Γ_ℓ with probability $1/2$ and to Γ_r with probability $1/2$. Let us fix a variable X and its side, say left. What happens with $n_X \cdot ||X||$ in (1) in the sum corresponding to left unblocked variables? If X is left blocked then, by Lemma 9, it will stay left blocked and so the contribution is and will be 0. If it is left unblocked, then its two first letters a, b are not new, so they are in Γ. If $S(X)$ has only those two letters, then with probability $1/2$ the a

will be in Γ_r and it will be popped and X will become left blocked (as $S(X)$ has only one letter), the same analysis applies, when the third leftmost letter is new. The remaining case is that the three leftmost letters in $S(X)$ are not new, let them be $a, b, c \in \Gamma$. By Lemma 4 $a \neq b \neq c$. With probability 1/4 $ab \in \Gamma_\ell\Gamma_r$ and with probability 1/4 $bc \in \Gamma_\ell\Gamma_r$. Those events are disjoint (as in one $b \in \Gamma_r$ and in the other $b \in \Gamma_\ell$) and so their union happens with probability 1/2. In both cases X will become left blocked, as a new letter is its first or second in $S(X)$. In all uninvestigated cases the contribution of $n_X \cdot ||X||$ cannot raise, which shows the claim in this case. The case of (3) is shown in the same way.

For (2), the analysis for an index i that is left unblocked is similar, but this time we consider the positions to the left of $\mathrm{Pos}_{\supseteq}(i)$ and $\mathrm{Pos}_{\supseteq}(i)$ can extend to them (instead of letters being popped from variables in case of (1)) and some of them may be compressed to one. Note that if there are no letters to the left/right then this index is blocked from this side. The case of (4) is shown in the same way. ◀

Space consumption. We now give the linear space bound on the size of equation. This formalises the intuition from the beginning of Section 3.2. As a first step, we show an upper-bound on the encoding size of the equation; define

$$H_d(U,V) = \sum_{i:\text{index}} ||U_0V_0[i]|| \cdot |\mathrm{Pos}_{\supseteq}(i)|, \quad H_n(U,V) = \sum_{i:\text{index}} 2|\mathrm{Pos}_{\supseteq}(i)| \cdot \log(|\mathrm{Pos}_{\supseteq}(i)|+1) \ ,$$

and $H(U,V) = H_d(U,V) + H_n(U,V)$. H_d corresponds to the size of $U_0V_0[I]$ in the encoding and H_d: of the numbers in the encoding.

▶ **Lemma 11.** *Given the equation (U,V) it holds that $||(U,V)|| \leq H_d(U,V) + H_n(U,V)$.*

The proof follows by simple symbolic transformation of the definitions.

Instead of showing a linear bound on $||(U,V)||$ we give a linear bound on $H(U,V)$. Recall that (U_0,V_0) denotes the input equation.

▶ **Lemma 12.** *Consider an equation $U = V$, its solution S, a phase of* LinWordEq *which makes the nondeterministic choices according to S and partitions according to the strategy. Let the returned equation be (U',V'). Then $H(U',V') \leq \frac{5}{6}H(U,V) + \alpha||(U_0,V_0)||$ and in a phase H on intermediate equations is at most $\beta H(U,V) + \gamma||(U_0,V_0)||$ for some constants α, β, γ.*

Proof. We separately estimate the H_d and H_n. Concerning H_d, let us first estimate $||U_0V_0[\mathrm{dep}(p)]||$ summed over positions p of letters popped into the equation during a phase (note, this does not include the size of numbers used in the encoding). For each variable we pop perhaps several letters to the left and right before block compression, but those letters are immediately replaced with single letters, so we count each as 1; also, when this side of a variable becomes blocked, it can pop at most one letter. Otherwise, a side of a variable pops at most 1 letter per pair compression, in which it is unblocked from this side. Note that the depint is the same as for variable, so the encoding size is $||X||$. So in total the bit-size of popped letters is at most:

$$\underbrace{\sum_{X\in\mathcal{X}} 2n_X \cdot ||X||}_{\text{block compression}} + \underbrace{\sum_{X\in\mathcal{X}} 2n_X \cdot ||X||}_{\text{after } X \text{ becomes blocked}} + \\ + \sum_{P:\text{ partition}} \left(\sum_{\substack{X\in\mathcal{X} \\ \text{left unblocked in } P}} n_X \cdot ||X|| + \sum_{\substack{X\in\mathcal{X} \\ \text{right unblocked in } P}} n_X \cdot ||X|| \right). \quad (5)$$

Observe that the third sum (the one summed over all partitions) at the beginning of the phase is equal to $\sum_X 2n_X \cdot ||X||$, as no side of the variable is blocked, and by the strategy point (1) its value at least halves every 4th pair compression (and it cannot increase, as by Lemma 9 no side of the variable can cease to be blocked). Thus (5) is at most

$$4\sum_X n_X \cdot ||X|| + 8\sum_X n_X \cdot ||X|| \left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right) = 20\sum_X n_X \cdot ||X|| \leq 20||(U_0, V_0)|| \ .$$

We now similarly estimate how many positions got into $\mathrm{Pos}_{\supseteq}(i)$ due to expansion of $\mathrm{Pos}_{\supseteq}(i)$: $\mathrm{Pos}_{\supseteq}(i)$ can expand to two letters during the block compression (to be more precise: to positions that are inside a block and to the positions to the let/right ones, but positions in a block are replaced with a single letter and one of them was in $\mathrm{Pos}_{\supseteq}(i)$) to one position at each side after i becomes blocked and by one position for each partition P in which this side of i is not blocked. So the increase in the bit-size is

$$\sum_{i\colon \text{index}} 4||U_0V_0[i]|| + \sum_{P\colon \text{partition}} \Bigg(\sum_{\substack{i\colon \text{index} \\ \text{left unblocked in } P}} ||U_0V_0[i]|| + \sum_{\substack{i\colon \text{index} \\ \text{right unblocked in } P}} ||U_0V_0[i]|| \Bigg) \tag{6}$$

and as in (5) similarly at the beginning of the phase the second sum (so the one summed by partitions) is $\sum_{i\colon \text{index}} 2||U_0V_0[i]|| = 2||(U_0, V_0)||$ and it at least halves every 4th partition, by strategy point (2). Thus similar calculations show that (6) is at most $20||(U_0, V_0)||$.

On the other hand, the number of positions in $\mathrm{Pos}_{\supseteq}(i)$ drops till the end of the phase by at least $\frac{|\mathrm{Pos}_{\supseteq}(i)|}{3} - 1$ due to compression:

1. If $U_0V_0[i]$ is a letter, then $\mathrm{Pos}_{\supseteq}(i)$ are all positions of letters and Lemma 2 yields that $\mathrm{Pos}_{\supseteq}(i)$ looses at least $\frac{|\mathrm{Pos}_{\supseteq}(i)|-1}{3}$ positions.
2. If $U_0V_0[i]$ is an ending marker, then the marker itself is unchanged and the remaining positions in $\mathrm{Pos}_{\supseteq}(i)$ are letter-positions and Lemma 2 applies to them, so $\mathrm{Pos}_{\supseteq}(i)$ looses at least $\frac{|\mathrm{Pos}_{\supseteq}(i)|-2}{3} < \frac{|\mathrm{Pos}_{\supseteq}(i)|}{3} - 1$ positions.
3. If $U_0V_0[i]$ is a variable then $\mathrm{Pos}_{\supseteq}(i)$ includes the position of a variable and Lemma 2 applies to strings of letters to the left and right, say of length ℓ, r, where $\ell + r = |\mathrm{Pos}_{\supseteq}(i)| - 1$.

Then due to compressions $\mathrm{Pos}_{\supseteq}(i)$ looses at least $\frac{\ell-1}{3} + \frac{r-1}{3} = \frac{|\mathrm{Pos}_{\supseteq}(i)|}{3} - 1$ positions.
Thus:

$$\begin{aligned} H_d(U', V') \leq\ & \underbrace{40||(U_0, V_0)||}_{\text{new positions in depints}} + \underbrace{\sum_{i\colon \text{index}} ||U_0V_0[i]|| \cdot \left(\frac{2}{3}|\mathrm{Pos}_{\supseteq}(i)| + 1\right)}_{\text{old positions lost}} \\ &= 40||(U_0, V_0)|| + \sum_{i\colon \text{index}} \frac{2}{3}||U_0V_0[i]|| \cdot |\mathrm{Pos}_{\supseteq}(i)| + \sum_{i\colon \text{index}} ||U_0V_0[i]|| \\ &= 41||(U_0, V_0)|| + \frac{2}{3}H_d(U, V) \ . \end{aligned}$$

We also estimate the maximal value of H_d during the phase, as for intermediate equations we cannot guarantee that the compression reduced the length of all letters. We already showed that in a phase we increase H_d by $40||(U_0, V_0)||$. This yields a bound of $H_d(U, V) + 40||(U_0, V_0)||$, which shows the part of the claim of Lemma for H_d.

Concerning H_n, for an index i let k_i, o_i, e_i denote, respectively: $|\mathrm{Pos}_{\supseteq}(i)|$ at the beginning of the phase, number of positions of letters popped from a variable with depint i and number of positions to whose depint i extended. First we estimate $\sum_{i\colon \text{index}} h(o_i)$ and $\sum_{i\colon \text{index}} h(e_i)$

and then use those estimations to calculate the bound on $H_n(U', V')$. We first inspect the case of o_i; let $P_1, P_2, \ldots$ denote the consecutive partitions in phase. We show that

$$\sum_{i\text{: index}} h(o_i) \leq \sum_{X\in\mathcal{X}} 25n_X + \sum_{m\geq 1} m \cdot \Big(\sum_{\substack{X\in\mathcal{X} \\ \text{left unblocked in } P_m}} n_X + \sum_{\substack{X\in\mathcal{X} \\ \text{right unblocked in } P_m}} n_X \Big). \quad (7)$$

The inequality follows as: if (one occurrence of) X popped o_X letters, then it was not blocked on left/right side for o_1/o_2 partitions, where $o_1 + o_2 \geq o_X - 4$ (note that one sequence can be popped to the left and right during block compression but it is immediately replaced with a single letter, so we treat them as one letter, also one letter can be popped to the left/right after X became blocked). Then in right hand side of (7) the contribution from (one occurrence of) X is at least

$$25 + \frac{o_1(o_1+1) + o_2(o_2+1)}{2} \geq \frac{(o_X-4)^2}{4} + \frac{o_X-4}{2} + 25 \geq o_X \log(o_X+1) \ ,$$

where the first inequality follows as $o_1 + o_2 \geq o_X - 4$ and the second can be checked by simple numerical calculation. Lastly, in (7) each o_i is equal to an appropriate o_X.

The sum in braces on the right hand side of (7) initially is at most $2|U_0V_0| \leq 2||(U_0, V_0)||$ and by strategy choice (3) it is at least halved every 4th step. So this sum is at most:

$$\sum_{i\geq 0} \underbrace{(16i+10)}_{4 \text{ consecutive steps}} \cdot \underbrace{2||(U_0,V_0)||}_{\text{initial size}} \cdot (1/2)^i = 104||(U_0, V_0)||$$

and consequently

$$\sum_{i\text{: index}} h(o_i) \leq 129||(U_0, V_0)|| \ . \quad (8)$$

The analysis for e_i is similar: for a single index i the estimation of the number of position by which $\mathrm{Pos}_{\supseteq}(i)$ extends is the same as the estimation of number of letters popped from an occurrence of a variable, thus

$$\sum_{i\text{: index}} h(e_i) \leq 129||(U_0, V_0)|| \ . \quad (9)$$

We now estimate, how many positions were lost due to compression, recall that k_i is the size of $\mathrm{Pos}_{\supseteq}(i)$ at the beginning of the phase. Using the same analysis as in the case of H_d, from Lemma 2 it follows that at least $\frac{k_i}{3} - 1$ positions were lost in the phase due to compression . Thus

$$H_n(U', V') \leq \sum_{i\text{: index}} h\left(\frac{2}{3}k_i + 1 + o_i + e_i\right). \quad (10)$$

Consider two subcases: if $\frac{2}{3}k_i + 1 + o_i + e_i \leq \frac{5}{6}k_i$, then the summand can be estimated as $h(\frac{5}{6}k_i) \leq \frac{5}{6}h(k_i)$ and we can upper bound the sum over those cases by $\frac{5}{6}\sum_{i\text{: index}} h(k_i)$. If $\frac{2}{3}k_i + 1 + o_i + e_i > \frac{5}{6}k_i$ then $1 + o_i + e_i > \frac{1}{6}k_i$ and so $\frac{2}{3}k_i + 1 + o_i + e_i < 5(1 + o_i + e_i)$. Thus (10) is upper-bounded by:

$$H_n(U', V') \leq \frac{5}{6} \sum_{i\text{: index}} h(k_i) + \sum_{i\text{: index}} h(5(1 + o_i + e_i)) \ .$$

Using simple properties of h as well as (8)–(9) we upper-bound the right hand side by $\frac{5}{6}H_n(U, V) + 15540||(U_0, V_0)||$.

We should estimate the maximal H_n value during the phase, as inside a phase we cannot guarantee that letters get compressed, i.e. estimate $\sum_{i:\text{ index}} h(k_i + o_i + e_i)$. Using similar calculation as in the case of (10) and properties of h we obtain:

$$\sum_{i:\text{ index}} h(k_i + o_i + e_i) \leq 8H_n(U,V) + 2064||(U_0,V_0)|| \ ,$$

which shows the claim of the Lemma in the case of H_n and so also in case of H. ◀

3.3 Proof of Theorem 3

By Lemma 1 all our subprocedures are sound, so we never accept an unsatisfiable equation.

We now give analyse the nondeterministic choices that yield termination, completeness and linear space consumption. Consider an equation $U = V$ at the beginning of the phase, let Γ be the set of letters in this equation. If it has a solution S', then it also has a solution S over Γ such that $|S(X)| = |S'(X)|$ for each variable: we can replace letters outside Γ with a fixed letter from Γ. During the phase we will make nondeterministic choices according to this S.

Let the equation obtained at the end of the phase be $U' = V'$ and S' be the corresponding solution. Then $|S'(U')| \leq \frac{2|S(U)|+1}{3}$ by Lemma 2 and we begin the next phase with S'. Hence we terminate after $\mathcal{O}(\log N)$ phases, where N is the length of some solution of the input equation.

Let the run also choose the partitions according to the strategy. We show by induction that for an equation (U,V) at the beginning of a phase $H(U,V) \leq \delta||(U_0,V_0)||$, where δ is a constant. Initially $H_n(U_0,V_0) = ||(U_0,V_0)||$ and $H_d(U_0,V_0) = 2||(U_0,V_0)||$, as for each index $\text{dep}(i) = \{i\}$; hence the claim holds. By Lemma 12 the inequality at the end of each phase holds for $\delta = 6\alpha$ for α from Lemma 12. For intermediate equations $H(U,V) \leq (6\alpha\gamma + \beta)||(U_0,V_0)||$, by Lemma 12, where β, γ are the constants from Lemma 12.

To upper-bound the space consumption, we also estimate other stored information: we also store the alphabet from the beginning of the phase (this is linear in the size of the equation at the beginning of the phase) and the mapping of this alphabet to the current symbols (linear in the equation at the beginning of the phase plus the size of the current equation). The terminating condition that some pair of letters in Γ^2 was not covered is guessed nondeterministically, we do not store Γ^2. The pair compression and block compression can be performed in linear space, see Lemma 6. Note that this includes the change of Huffman coding.

References

1 Volker Diekert, Claudio Gutiérrez, and Christian Hagenah. The existential theory of equations with rational constraints in free groups is PSPACE-complete. *Inf. Comput.*, 202(2):105–140, 2005.

2 Volker Diekert, Artur Jeż, and Manfred Kufleitner. Solutions of word equations over partially commutative structures. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP*, volume 55 of *LIPIcs*, pages 127:1–127:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.127.

3 Volker Diekert, Artur Jeż, and Wojciech Plandowski. Finding all solutions of equations in free groups and monoids with involution. *Inf. Comput.*, 251:263–286, 2016. doi:10.1016/j.ic.2016.09.009.

4 Volker Diekert and Markus Lohrey. Word equations over graph products. *International Journal of Algebra and Computation*, 18(3):493–533, 2008.

5 Claudio Gutiérrez. Satisfiability of word equations with constants is in exponential space. In *FOCS*, pages 112–119. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743434.

6 Joxan Jaffar. Minimal and complete word unification. *J. ACM*, 37(1):47–85, 1990.

7 Artur Jeż. Context unification is in PSPACE. In Elias Koutsoupias, Javier Esparza, and Pierre Fraigniaud, editors, *ICALP*, volume 8573 of *LNCS*, pages 244–255. Springer, 2014. doi:10.1007/978-3-662-43951-7_21.

8 Artur Jeż. Recompression: a simple and powerful technique for word equations. *J. ACM*, 63(1):4:1–4:51, Mar 2016. doi:10.1145/2743014.

9 Antoni Kościelski and Leszek Pacholski. Complexity of Makanin's algorithm. *J. ACM*, 43(4):670–684, 1996.

10 Gennadiĭ Makanin. The problem of solvability of equations in a free semigroup. *Matematicheskii Sbornik*, 2(103):147–236, 1977. (in Russian).

11 Gennadiĭ Makanin. Equations in a free group. *Izv. Akad. Nauk SSR*, Ser. Math. 46:1199–1273, 1983. English transl. in Math. USSR Izv. 21 (1983).

12 Yuri Matiyasevich. Some decision problems for traces. In Sergej Adian and Anil Nerode, editors, *LFCS*, volume 1234 of *LNCS*, pages 248–257. Springer, 1997. Invited lecture.

13 Wojciech Plandowski. Satisfiability of word equations with constants is in NEXPTIME. In *STOC*, pages 721–725. ACM, 1999. doi:10.1145/301250.301443.

14 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3):483–496, 2004. doi:10.1145/990308.990312.

15 Wojciech Plandowski. An efficient algorithm for solving word equations. In Jon M. Kleinberg, editor, *STOC*, pages 467–476. ACM, 2006. doi:10.1145/1132516.1132584.

16 Wojciech Plandowski and Wojciech Rytter. Application of Lempel-Ziv encodings to the solution of word equations. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *ICALP*, volume 1443 of *LNCS*, pages 731–742. Springer, 1998. doi:10.1007/BFb0055097.

17 Alexander A. Razborov. *On Systems of Equations in Free Groups*. PhD thesis, Steklov Institute of Mathematics, 1987. In Russian.

18 Klaus U. Schulz. Makanin's algorithm for word equations – two improvements and a generalization. In Klaus U. Schulz, editor, *IWWERT*, volume 572 of *LNCS*, pages 85–150. Springer, 1990. doi:10.1007/3-540-55124-7_4.

Solutions of Twisted Word Equations, EDT0L Languages, and Context-Free Groups*†

Volker Diekert[1] and Murray Elder[2]

1 **Universität Stuttgart, Formal Methods in CS, Stuttgart, Germany**
`diekert@fmi.uni-stuttgart.de`
2 **University of Technology Sydney, Sydney, Australia**
`murray.elder@uts.edu.au`

Abstract

We prove that the full solution set of a twisted word equation with regular constraints is an EDT0L language. It follows that the set of solutions to equations with rational constraints in a context-free group (= finitely generated virtually free group) in reduced normal forms is EDT0L. We can also decide whether or not the solution set is finite, which was an open problem. Moreover, this can all be done in PSPACE. Our results generalize the work by Lohrey and Sénizergues (ICALP 2006) and Dahmani and Guirardel (J. of Topology 2010) with respect to complexity and with respect to expressive power. Both papers show that satisfiability is decidable, but neither gave any concrete complexity bound. Our results concern all solutions, and give, in some sense, the "optimal" formal language characterization.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.4.2 Grammars and Other Rewriting Systems, F.4.3 Formal Languages

Keywords and phrases Twisted word equation, EDT0L, virtually free group, context-free group

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.96

1 Introduction

In a seminal paper [21] Makanin showed that the problem *WordEquations* is decidable. The first complexity estimation of that problem was a tower of several exponential functions, but this dropped down to PSPACE by Plandowski [24] using compression. The insight that long solutions of word equations can be efficiently compressed is due to [25], which led to the conjecture that WordEquations is NP-complete. In 2013 Jeż applied his *recompression* technique: he presented a new and simple NSPACE($n \log n$) algorithm to solve word equations [16]. (Very recently, he lowered the complexity to NSPACE(n) [17]). Actually his method achieved more: it describes all solutions, copes with rational constraints (which is essential in applications), and it extends to free groups [6]. Building on ideas in [6], Ciobanu and the present authors showed that the full solution set of a given word equation with rational constraints is EDT0L [3]. This was known before only for quadratic word equations by [11]. *EDT0L-languages* are defined by a certain type of Lindenmayer system, see [27]. The motivation for [3] was to prove that the full solution set in reduced words of equations in free groups is an indexed language, an open problem at the time [12, 15]. But EDT0L is better: it is strictly included in the class of indexed languages [9].

* A full version of the paper is available at `https://arxiv.org/abs/1701.03297`.

† Research supported by Australian Research Council (ARC) Project DP 160100486 and German Research Foundation (DFG) Project DI 435/7-1

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 96; pp. 96:1–96:14

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

The class of finitely generated (f.g. for short) virtually free groups arises in many different ways. A fundamental theorem of Muller and Schupp (relying on [8]) says that a f.g. group is virtually free if and only if it is *context-free* [23]. This means that, given any set of monoid generators A, the set of words $w \in A^*$ which represent $1 \in V$ forms a context-free language. Other characterizations include: (1) fundamental groups of finite graphs of finite groups [18], (2) f.g. groups having a Cayley graph with finite treewidth [19], (3) groups having a finite presentation by some geodesic string rewriting system [13], and (4) f.g. groups having a Cayley graph with decidable monadic second-order theory [19], etc. See [7]. We show that given a f.g. virtually free group V there is a PSPACE-algorithm which produces, for a given equation with rational constraints, an EDT0L grammar which describes the full solution set in reduced words over a natural set of generators. Several remarks are in order here. First, virtually free groups (which are not free) have torsion, and this is serious obstacle to applying the techniques used in [24, 16, 6, 3]. A driving motivation to study virtually free groups is the connection to *word hyperbolic groups* [14]. Solving equations in torsion-free hyperbolic groups reduces to solving equations in free groups [26], but solving equations in word hyperbolic groups with torsion reduces to solving equations in virtually free groups which in turn reduces to solving "twisted" word equations with rational constraints [4]. The question how to solve "twisted" word equations was asked by Makanin ([22, Problem 10.26(b)]) and solved by Lohrey and Sénizergues [20] and Dahmani and Guirardel [4]. Both papers show more general results, and yield independent proofs that satisfiability for equations over a f.g. virtually free group is decidable. The approach in [4] assumes a bound on the so-called "exponent of periodicity", thus it does not handle the full set of solutions. Lohrey and Sénizergues [20] prove a general transfer result which applies to all solutions, but this does not produce any "nice" description. Note that to have "some description" of all solutions is not enough to decide finiteness, in general. Our EDT0L description pays attention that every solution is represented exactly once. The other achievement here is a first known concrete complexity bound: PSPACE, a surprisingly low complexity given the circumstances.

Therefore, the present paper extends [4, 20] in various aspects. As in [4] we are working over twisted word equations with rational constraints, which is the natural approach due to Bass-Serre theory [31], see [18] (and [29, 30] for effective constructions). Our main new contribution is within combinatorics on words. Although we follow the general scheme [16, 6, 3] to define a sound and complete algorithm to produce an NFA describing all solutions, the technical details are quite far from previous methods.

Proofs omitted from the present paper can be found in [5].

1.1 Preliminaries

An *alphabet* is a finite set of *letters*; and Σ^* denotes the free monoid of *words* over Σ. The empty word is denoted by 1. The length $w \in \Sigma^*$ is $|w|$, and $|w|_a$ counts how often a letter a appears in w. Let M be any monoid. Then $u \in M$ is a *factor* of $v \in M$ if we can write $v = xuy$ for some x, y. If $x = 1$ (resp. $y = 1$), then we say that u is a *prefix* (resp. *suffix*) of v. For a prefix, we also write $u \leq v$. An *involution* is a bijection $x \mapsto \overline{x}$ such that $\overline{\overline{x}} = x$ for all x in the set. A *monoid with involution* additionally has to satisfy $\overline{xy} = \overline{y}\,\overline{x}$. If G is a group, then it is a monoid with involution by taking $\overline{g} = g^{-1}$ for all $g \in G$. Thus, we identify $\overline{g}$ and g^{-1} in groups. In the following, every alphabet comes with an involution. This is no restriction since the identity is always an involution for sets. A *morphism* between sets with involution is a mapping respecting the involution. A *morphism* between monoids with involution is a homomorphism $\varphi : M \to N$ such that $\varphi(\overline{x}) = \overline{\varphi(x)}$. For $\Delta \subseteq M \cap N$ we say that it is a Δ-*morphism* if $\varphi(x) = x$ for all $x \in \Delta$. A bijective morphism from a set to itself is called an *automorphism* and the set of automorphisms on a set (or monoid)

M forms the group $\mathrm{Aut}(M)$. Let G be a group. It acts on a set (with involution) X by a mapping $x \mapsto g \cdot x$ if $1 \cdot x = x$, $f \cdot (g \cdot x) = (fg) \cdot x$ (and $f \cdot \overline{x} = \overline{f \cdot x}$). If G acts on a monoid (with involution) M, then we additionally demand that every group element acts as an automorphism: $f \cdot (xy) = (f \cdot x)(f \cdot y)$. Frequently, we write $f(x)$ instead of $f \cdot x$. The specification of regular constraints is given here by assigning to each constant and variable an element in a finite monoid (typically the finite monoid is a monoid of Boolean matrices and arises as the transformation monoid of a finite automaton.) By making the finite monoid larger, we can turn it into a monoid N with involution and where G acts on it. This allows us to represent regular constraints using a morphism $\mu : (A \cup (G \times \mathcal{X}))^* \to N$ which respects the involution and the action of G. In the following we fix the finite monoid N and we assume that all morphisms to N respect the involution and G action. We say that M is an *NG-i-monoid* if M is a monoid with involution and a G action together with a morphism $\mu : M \to N$. (In this abbreviation the i stands for "involution".) If not explicitly stated otherwise all monoids under consideration are NG-i-monoids (including N itself). A morphism between NG-i-monoids M, M' with morphisms μ, μ' is a morphism $\varphi : M \to M'$ such that $\varphi(g \cdot x) = g \cdot (\varphi(x))$ and $\mu'\varphi = \mu$. Henceforth, by default, a morphism means a morphism between NG-i-monoids.

Regular languages in finitely generated free monoids can be defined via nondeterministic finite automata (NFA for short) or via recognizability via homomorphisms to finite monoids, to mention just two possible definitions. This notion extends to every monoid M: an NFA is a directed finite graph $\mathcal{A}$ with initial and final *states*, where the transitions are labeled with elements of the monoid M. A transition labeled by $1 \in M$ is called an *ε-transition*. We say that $m \in M$ is *accepted* by the automaton $\mathcal{A}$ if there exists a path from some initial to some final state such that multiplying the edge labels together yields m. This defines the accepted language $L(\mathcal{A}) = \{m \in M \mid m \text{ is accepted by } \mathcal{A}\}$. According to [10] a subset $L \subseteq M$ is *rational* if and only if L is accepted by some NFA over M. An NFA is called *trim* if every state is on some path from an initial to a final state. Ensuring the NFA that we construct in our proof below is trim, allows us to decide emptiness or finiteness of the solution set.

A subset $L \subseteq A^* \times \cdots \times A^*$ is called *EDT0L* if there some (extended) alphabet C with $c_1, \ldots, c_k \in C$ such that $A \subseteq C$ and a rational set $\mathcal{R} \subseteq \mathrm{End}(C^*)$ of endomorphisms over C^* such that $L = \{(h(c_1), \ldots, h(c_k)) \mid h \in \mathcal{R}\}$. The classical definition for EDT0L refers to $k = 1$. Our definition uses a characterization of EDT0L languages due to Asveld [1, 28].

Let B and $\mathcal{Y}$ be two disjoint NG-i-alphabets. We call B the alphabet of *constants* and $\mathcal{Y}$ the set of *twisted variables*. It is convenient to choose a minimal subset $\mathcal{X} \subseteq \mathcal{Y}$ such that every $Y \in \mathcal{Y}$ has the form $Y = f \cdot X$ for some $X \in \mathcal{X}$ and $f \in G$. Moreover, we assume $X \neq \overline{X}$ for all variables. If G acts without fixed points on $\mathcal{Y}$, then we identify $\mathcal{Y} = G \times \mathcal{X}$ and the action becomes $g \cdot (f, X) = (gf, X)$. By $M(B, \mathcal{X}, \theta, \mu)$ we denote an NG-i-monoid which is generated by $B \cup \{f(X) \mid f \in G, X \in \mathcal{X}\}$ together with a finite set θ of *homogeneous* defining relations, which means every $(x, y) \in \theta$ satisfies $|x| = |y|$. We always assume that $(x, y) \in \theta$ implies $\mu(x) = \mu(y)$, $(\overline{x}, \overline{y}) \in \theta$, and $(f(x), f(y)) \in \theta$ for all $f \in G$, even if these relations are not listed in the specification of θ. For complexity issues we require $|x| \leq 2$ for each $(x, y) \in \theta$ and $|\theta| \in \mathcal{O}(|G| \, \|\mathcal{S}\|^2)$ where $\|\mathcal{S}\|$ is specified in Theorem 1. The homogeneity condition makes it possible to solve the word problem and all other computational issues for the quotient $M(B, \mathcal{X}, \theta, \mu) = M(B, \mathcal{X}, \emptyset, \mu)/\{x = y \mid (x, y) \in \theta\}$ within our space bound.

2 The main results

Let A an alphabet of constants and G be a subgroup of $\mathrm{Aut}(A)$. Initially, the set of twisted variables is $G \times \mathcal{V}$. For a word $w \in A^*$ and $f \in G$ we use the notation $f(w) = (f, w)$;

and we hence identify $(A \cup (G \times \mathcal{V}))^* = ((G \times (A^* \cup \mathcal{V}))^*$. We abbreviate $(1, x)$ as x for $x \in A^* \cup \mathcal{V}$. A system $\mathcal{S}$ of *twisted word equations with rational constraints* is given by a set of pairs $\{(U_i, V_i) \mid 1 \leq i \leq s\}$ where $U_i, V_i \in (A \cup (G \times \mathcal{V}))^*$ are *twisted words* and a morphism $\mu_0 : (A \cup (G \times \mathcal{V}))^* \to N$. It is specified by its restriction to $A \cup \mathcal{V}$; and μ_0 respects the involution and the action of G.

As usual, a twisted equation (U_i, V_i) is also written as $U_i = V_i$. A *solution* of $\mathcal{S}$ is given a morphism $\sigma : \mathcal{V} \to A^*$ which is (uniquely) extended to an A-morphism of NG-i-monoids $\sigma : (A \cup (G \times \mathcal{V}))^* \to A^*$ such that $\sigma(U_i) = \sigma(V_i)$ for all i and $\mu_0\sigma(X) = \mu_0(X)$ for all variables. Hence, $\mu_0\sigma = \mu_0$. The full *solution set* $\mathrm{Sol}(\mathcal{S})$ for $\mathcal{V} = \left\{X_1, \overline{X_1}, \ldots, X_k, \overline{X_k}\right\}$ is $\mathrm{Sol}(\mathcal{S}) = \{(\sigma(X_1), \ldots, \sigma(X_k)) \in A^* \times \cdots \times A^* \mid \sigma \text{ solves } \mathcal{S}\}$. We define the size $\|\mathcal{S}\|$ by $\|\mathcal{S}\| = |G| + |A| + |\mathcal{V}| + s + \sum_{1 \leq i \leq s} |U_i V_i|$.

Convention. For better readability we don't measure N, but we add the general hypotheses that N is given in such a way that the specification and all necessary computations over N (multiplication, computing the involution and the G action) can be done in polynomial space with respect to $\|\mathcal{S}\|$. This is no restriction, as we can add trivial equations to enlarge $\|\mathcal{S}\|$.

▶ **Theorem 1.** *There is a* PSPACE *algorithm which takes as input a system of twisted word equations with rational constraints $\mathcal{S}$ as above with input size $\|\mathcal{S}\|$. The output is an extended alphabet C of size $\mathcal{O}(|G|^2 \|\mathcal{S}\|^2)$, letters $c_X \in C$ for each $X \in \mathcal{V}$, and a trim NFA $\mathcal{A}$ accepting a rational set of A-morphisms $L(\mathcal{A}) \subseteq \mathrm{End}(C^*)$ such that*

$$\mathrm{Sol}(\mathcal{S}) = \left\{(h(c_{X_1}), \ldots, h(c_{X_{|\mathcal{V}|}})) \in C^* \times \cdots \times C^* \;\middle|\; h \in L(\mathcal{A})\right\}. \tag{1}$$

Intermediate equations, which label states of the NFA, have a length bound in $\mathcal{O}(|G| \|\mathcal{S}\|^2)$. Moreover, $\mathrm{Sol}(\mathcal{S}) = \emptyset$ if and only if $L(\mathcal{A}) = \emptyset$, and $|\mathrm{Sol}(\mathcal{S})| < \infty$ if and only if $\mathcal{A}$ doesn't contain any directed cycle.

The result above is far-reaching extension of Makanin's classical result on pure word equations. It combines combinatorics on words, automata theory, formal languages, and group actions on alphabets. It doesn't use band complexes, Makanin-Razborov diagrams or results from algebraic geometry over groups [4, 2]. Here, a virtually free group V is given by a group extension of a free group $F(B)$ with a finite group G with the natural set $A = B \cup B^{-1} \cup G \setminus \{1\}$ as generators. We represent elements of V by *reduced normal forms* in $\widehat{V}$, where $\widehat{V}$ is the set of words in $B^*G \subseteq A^*$ without factors $b\overline{b}$. Thus, we have a natural notion of *solution in reduced normal forms.*

▶ **Corollary 2.** *Let V be a f.g. virtually free group. There is an* NSPACE$(m^2\|\mathcal{S}\|^2 \log(\|\mathcal{S}\|))$ *algorithm such that:*

Input. *A system $\mathcal{S}$ of s equations $U_i = V_i$ over V with rational constraints and in variables $X_1, \ldots, X_k$, where $\|\mathcal{S}\| = k + \sum_{1 \leq i \leq s} |U_i V_i|$ and m denotes the number of states for the NFA's to encode constraints.*

Output. *An extended alphabet C of size $\mathcal{O}(\|\mathcal{S}\|^2)$, letters $c_X \in C$ for each variable, and a trim NFA $\mathcal{A}$ accepting a rational set of A-morphisms over C^* such that*

$$\left\{(h(c_{X_1}), \ldots, h(c_{X_k})) \in (C^*)^k \;\middle|\; h \in L(\mathcal{A})\right\} = \left\{(\sigma(X_1), \ldots, \sigma(X_k)) \in \widehat{V}^k \;\middle|\; \sigma \text{ solves } \mathcal{S}\right\}.$$

Moreover, there is no solution if and only if $L(\mathcal{A}) = \emptyset$, and there are infinitely many solutions if and only if $\mathcal{A}$ contains a directed cycle.

The reduction of Corollary 2 to Theorem 1 follows [4] very closely, see [5] for details:

1. Embed V into a semi-direct product $F(S) \rtimes G$ using Bass-Serre theory. This encodes $\widehat{V}$ as a rational set in S^*G and allows us to view a system of equations over V (with rational constraints) as a system of twisted word equations with rational constraints over S.
2. Handling of rational constraints by transformations on NFA's by standard methods.
3. Projection of EDT0L languages and respecting reduced normal forms using the fact that the embedding satisfies $B \subseteq S$.

3 Outline of the proof of Theorem 1

The actual proof of Theorem 1 is rather technical, so this extended abstract outlines the central ideas only. The focus is on those parts which are original and where the twisting forces us to deviate from what has been done elsewhere. Jeż's recompression technique is based on two procedures: *block-compression* and *pair-compression*; solutions are obtained by iteratively *popping* the first and last letters of variables (performing moves of the form $X \mapsto aX$), which increases the length of the equation, and *compressing* factors by replacing pairs ab and powers a^λ by a single (new) letter. In the "untwisted" setting, when we compress a pair ab we replace every occurrence of the factor that is "visible" in the equation, but in the twisted case, the pair ab appearing on one side of the equation needs to match with $f(ab)$ on the other side, which causes complications. The basic problem is that twisting of a word $(ab)^\lambda$ by some $f \in G$ may result in $f(ab)^\lambda = (ba)^\lambda$. The complications related to this will become clear below. Therefore we introduce two new procedures. First we define a new and more general δ-*periodic-compression* w.r.t. some $\delta \in \Theta(|G|\,\|\mathcal{S}\|)$. In some sense, δ-periodic-compression removes the problem caused by $f(p)^\lambda = q^\lambda$ where p and q are primitive words of length at most δ. (Powers of long primitive words are then handled in later iterations.) Performing one δ-periodic-compression will result in a situation where "equivalent" positions in the solution are far apart. This property is used for our version of pair-compression without the possibility to "uncross" pairs as is the usual strategy. Instead we do the following. First we pop out from every $\sigma(X)$ rather long prefixes and suffixes. After that we find room to compress enough pairs ab into fresh letters c. We cannot compress pairs by their label (twisting prevents that), so we compress only pairs ab where the corresponding two positions have no equivalent position which is located at some border of an occurrence of a variable. This is our leads to a new definition of *twisted-pair-compression*. Of course, we must define precisely when positions are equivalent and everything must take the action of G and rational constraints into account. Last but not least, we must realize the procedure by following arcs in an NFA where the labels are endomorphisms over some extended alphabet C. This yields the EDT0L property of the full solution set, more importantly it transforms questions about solvability of equations into structural properties of a finite graph.

One of the new features presented here is the ambient algebraic structure: in the case of free monoids (resp. free groups) the intermediate monoids were partially commutative. Twisting leads to more complicated defining relations. More concretely, when working with an equation $U = V$ over constants B and variables $\mathcal{X}$ with constraints defined by an NG-i-monoid morphism $\mu : B \cup \mathcal{X} \to N$, we deal with an NG-i-monoid denoted by $M(B, \mathcal{X}, \theta, \mu)$. The algebraic structure is a quotient monoid $(B \cup \mathcal{X})^* / \{xy = zx \mid (xy, zx) \in \theta\}$, where $x \in B \cup \mathcal{X}$ and $y, z \in (B \cup \mathcal{X})^*$ with $|y| = |z|$. The idea is that, reading a word in $w \in (B \cup \mathcal{X})^*$, the position of x is not fixed, it "floats" by conjugating y to z or vice versa, without changing the length of W. This possibility of "floating" is essential in our approach.

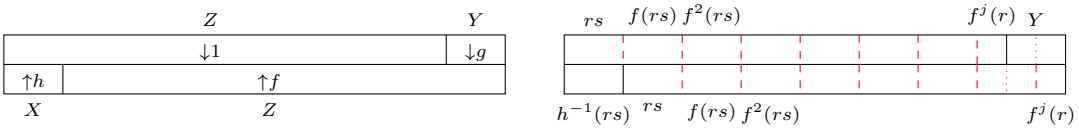


Figure 1 "Graphical" proof of Proposition 3.

3.1 States of the NFA

We start with a system of $\mathcal{S}$ of s equations $U_i = V_i$ over some alphabet A of constants and in variables X_j. We encode $\mathcal{S}$ as a single word using a marker symbol and we obtain the *initial equation* as:

$$W_{\text{init}} = \#X_1\# \cdots \#X_k\#U_1\# \cdots U_s\# \ \overline{\#X_1\# \cdots \#X_k\#V_1\# \cdots V_s\#}. \tag{2}$$

Note that $\sigma(W) = \sigma(\overline{W})$ if and only if $\sigma(U_i) = \sigma(V_i)$ for all i. We fix $n = |W_{\text{init}}|$. Note that this implies $n > |A| + |\mathcal{V}|$ and $\|\mathcal{S}\| \in |G| + \Theta(n)$. States of the transition system are denoted as $(W, B, \mathcal{X}, \theta, \mu)$. We call a state an *extended equation*. Here, B are the current constants and $\mathcal{X}$ are the current variables with $A \subseteq B \subseteq C$ and $\mathcal{X} \subseteq \Omega$ where C and Ω are fixed and of size $\mathcal{O}(|G|^2\|\mathcal{S}\|^2)$ and $W \in M(B, \mathcal{X}, \theta, \mu)$ has length bounded by $\mathcal{O}(|G|\|\mathcal{S}\|^2)$. A solution is a morphism (of NG-i-monoids) $\sigma : M(B, \mathcal{X}, \theta, \mu) \to M(B, \theta, \mu)$ such that $\sigma(W) = \sigma(\overline{W})$. Here, $M(B, \theta, \mu)$ is the submonoid of $M(B, \mathcal{X}, \theta, \mu)$. If θ is empty, then we speak about a *standard state*. We begin at a standard state and the aim is to track for every solution a path from the initial standard state $(W_{\text{init}}, A, G \times \mathcal{V}, \emptyset, \mu_0)$ to some final state $(W, B, \emptyset, \emptyset, \mu)$ without types and variables such that $W = \overline{W}$.

We need to reuse names for constants, so we also need a procedure, called *alphabet-reduction*, to get rid of *invisible* constants. These are letters $b \in B$ where for no $f \in G$ the letter $f(b)$ appears in W. Since a given solution σ might use them, we cannot simply throw them out. This forces us to consider *entire solutions* which are pairs (α, σ) where σ is a solution as above and $\alpha : M(B, \theta, \mu) \to A^*$ is an A-morphism.

3.2 Twisted conjugacy

An important concept in our approach is *twisted conjugacy*. We say that words $x, y \in A^*$ are *twisted conjugate* if there are $f, g, h \in G$ and $z \in A^*$ such that $zg(y) = h(x)f(z)$.

▶ **Proposition 3.** *Let σ be a solution of $Z(g, Y) = (h, X)(f, Z)$ such that $|\sigma(X)|$ satisfies $1 \leq |\sigma(X)| < |\sigma(Z)|$. Then there are words $r \in A^+$, $s \in A^*$ and $e, j \in \mathbb{N}$ with $0 \leq j < |G|$ such that $\sigma(X) = h^{-1}(rs)$ and $\sigma(Z) = ((rs)f(rs) \cdots f^{|G|-1}(rs))^e \, f^0(rs) \cdots f^{j-1}(rs)f^j(r)$.*

3.3 δ-periodic-compression

Recall that $w = a_1 \cdots a_n$ with $a \in A$ has *period* $p \in \mathbb{N}$ if $a_i = a_{i+p}$ for all $1 \leq i \leq n - p$. Let δ be some positive natural number. We say that a word w is *δ-periodic* if it has some period less or equal to than δ. Let u be a prefix (resp. factor, resp. suffix) of some nonempty word w. We say that u is a *maximal δ-periodic prefix (resp. factor, resp. suffix)* in w if we cannot extend the occurrence of the factor u inside w by any letter to the right or left, to get a δ-periodic word. A δ-periodic word w is called *long* if $|w| \geq 3\delta$, and *very long* if $|w| \geq 10\delta$. Standard knowledge in combinatorics on words tells us:

▶ **Lemma 4.** *Let w be a δ-periodic word and $w = p^e r = q^f s$ such that p, q are primitive, $|p| \le |q| \le \delta$, $1 \ne r \le p$, $1 \ne s \le q$, and $|w| \ge 2\delta$. Then $p = q$, $e = f \ge 1$, and $r = s$.*

Let us give a high-level description of our new procedure δ-periodic-compression. For simplicity, we deal just with a single "triangulated" twisted equation $(f, X)w(g, Y) = Z$ where X, Y, Z are variables and $w \in B^*$ is word over the current constants B. We consider a fixed solution σ and we ignore rational constraints by assuming $N = \{1\}$. Moreover, we assume that for every letter $b \in B$ there is some $f \in G$ such that $f(b)$ is a letter in w. Thus, we start with an alphabet-reduction which removes invisible letters for a given solution. Since σ is a solution, we can identify positions in w with positions in $\sigma(Z)$. These identified positions carry the same label and we also say that these positions are *visible*.

Let us consider all very long maximal δ-periodic factors $q^d q'$, written as $up^e rv$, of $\sigma(Z)$ which have an occurrence with a visible position. Note that their total number is bounded by $|w|/\delta$. In the description we assume that $|u| = |v| = 3\delta$, p is primitive of length at most δ and $1 \ne r \le p$. Hence, $up^e rv$ defines the triple (p, r, e) uniquely by Lemma 4.

The idea is that at the end we arrive at a state with a solution where all occurrences of these factors $up^{e_\lambda} rv$ are replaced by $u[r, s, \lambda]v$ where $[r, s, \lambda]$ is the notation for a single fresh letter and $rs = p$. Here λ is a formal symbol taken from some some index set Λ of size at most $|w|/\delta$. In order to avoid many case distinctions we consider the following (in some sense most interesting) special case, only. We assume that $\sigma(X)$ is a very long periodic word, $\sigma(Y)$ has a very long δ-periodic prefix, and $\sigma(Z)$ has a δ-periodic prefix longer than $|\sigma(X)|$, but no long δ-periodic suffix. Moreover, we assume that w has more than two very long δ-periodic factors. Note that $up^{e_\lambda} rv = urq^{e_\lambda} v$ if $1 \ne r \ne p$, $p = rs$, and $q = sr$. Let us resume: let $u_\lambda p_\lambda^{e_\lambda} r_\lambda v_\lambda$ be an occurrence of a very long δ-periodic factor in $\sigma(Z)$ with at least one visible position, $|u_\lambda| = |v_\lambda| = 3\delta$, and p_λ is primitive with $|p_\lambda| \le \delta$. Thus, $\lambda \in \Lambda$. There are three cases which we distinguish by using the names $\lambda, \nu, \rho \in \Lambda$. First, the occurrence of $u_\lambda p_\lambda^{e_\lambda} r_\lambda v_\lambda$ is the δ-periodic prefix of $\sigma(Z)$. As, by our simplification assumption, this prefix is longer than $\sigma(X)$, we deduce that we can write $\sigma(f, X) = u_\lambda p_\lambda^e p'$ with $p' \le p_\lambda$. Second, all "inner" positions $p_\nu^{e_\nu} r_\nu$ of $z = u_\nu p_\nu^{e_\nu} r_\nu v_\nu$ are visible. In this case, since $\sigma(X)$ (resp. $\sigma(Y)$) has a very long prefix (resp. suffix), this corresponds to an occurrence of the factor z in w. Third we can write $\sigma(g, Y) = p'' p_\rho^{e'} r_\rho v_\rho y$ with $e' \ge 6$ and $p'' \le p_\rho$ for some maximal δ-periodic factor $u_\rho p_\rho^{e_\rho} r_\rho v_\rho$ of $\sigma(Z)$ with $\rho \in \Lambda$. Moreover, we are in the case that the maximal δ-periodic prefix of $v_\rho y$ is v_ρ and $y \ne 1$. As we assumed that w has more than two very long δ-periodic factors, we can write $w = w_1 p_\nu^\nu r_\nu w_2$.

The procedure introduces at this point (for each $\lambda \in \Lambda$) new "typed" variables: $[X, p_\lambda]$, $[Y, \overline{p_\lambda}]$, and $[Z, p_\lambda]$. Actually, we need many more variables. Whenever we introduce variable $[V, p]$ we also introduce $[\overline{V}, \overline{p}]$ and $[V, qa]$ for $p = aq$; and we iterate this process. Finally, the action of G is defined by identifying $(f, [V, p])$ with $(1, [V, f(p)]) = [V, f(p)]$. (Note that $(f, [V, p]) = (g, [V, p]) \iff f(p) = g(p)$.)

The maximal number of these typed variables introduced by any equation with at most n variables is at most $2n|G|\delta$. The factor $2n$ is there because we consider for every variable a prefix and a suffix; and the factor δ comes in because $|p| \le \delta$ and with $p = aq$ every conjugate qa is also considered. Let $\mathcal{X}'$ be the enlarged set of untyped variables $X \in \mathcal{X}$ and fresh typed variables $[V, p]$. Together with introducing these variables we switch to the algebraic structure to read the equation in the monoid $M(B, \mathcal{X}', \theta, \mu')$ where the defining relations are given by $\theta = \{(a[V, p], [V, q]a \mid ap = qa \wedge a \in B\}$. We define the *type* of $[V, p]$ to be $\theta([V, p]) = p$ and we observe that the defining relations imply $p[V, p] = [V, p]p$ in $M(B, \mathcal{X}', \theta, \mu')$. We need a stronger notion of *solution* for typed variables in order to prevent that an unsolvable equation is transformed in a solvable one. If $\theta([V, p]) = p$, then we require $\sigma([V, p]) \in p^*$. Since σ is

a morphism, it also satisfies the defining relations. Hence, $a\sigma([V,p]) = \sigma([V,q])a$ implies $|\sigma([V,p])| = |\sigma([V,p])|$, too. The value of $\mu'([V,p])$ is defined implicitly in the following loop.

The loop is over all variables in some order. Of course, whatever happens to a variable V forces a simultaneous change in $\overline{V}$, too. We pop from each variable the maximal δ-periodic suffix of $\sigma(X)$ if this suffix is longer than 3δ. Otherwise we do nothing. As we have no control on the length of this suffix, we introduce a new typed variable. (Clearly, as we consider X and $\overline{X}$ prefixes and suffixes are popped out, and each X may produce two typed variable.) What we do in our concrete situation (where we have $\Lambda = \{\lambda, \nu, \rho\}$) is the following:

1. We substitute (f, X) by $\tau(f, X) = u_\lambda[X, p_\lambda]p_\lambda^\ell p'$. (So, X vanishes.) Moreover, for technical reasons, we require $5\delta < |p_\lambda^\ell p'| \le 6\delta$. We can define $\sigma'([X, p_\lambda]) \in p_\lambda^*$ such that $\sigma(f, X) = u_\lambda\sigma'([X, p_\lambda])p_\lambda^\ell p'$.
2. We substitute (g, Y) by $\tau(g, Y) = p''p_\rho^r[Y, p_\rho](g, Y)$ with the length condition $5\delta < \left|p''p_\rho^r\right| \le 6\delta$. We can define $\sigma'(g, Y) = v_\rho y$ and $\sigma'([Y, p_\rho]) \in p_\rho^*$ such that $\sigma(g, Y) = p''p_\rho^r\sigma'([Y, p_\rho])\sigma'(g, Y)$.
3. We substitute Z by $\tau(Z) = sq^{\ell'}[Z, q]Z$ with the length condition $5\delta < |sq^{\ell'}| \le 6\delta$. Here q is the conjugate of p_λ such that $q = sr_\lambda$, $p_\lambda = r_\lambda s$. We can define $\sigma'(Z) = v_\lambda z$ and $\sigma'([Z, q]) \in q^*$ with $sq^{\ell'}\sigma'([Z, q])\sigma'(Z) = \sigma(Z)$.

This leads to a new solution σ' to the twisted equation $u_\lambda[X, p_\lambda]p_\lambda^\ell p'wp''p_\rho^{r'}[Y, p_\rho](g, Y) = sq^{\ell'}[Z, q]Z$. We rename σ' as σ. Note that u_λ is a prefix of $sq^{\ell'}$. The positions of $[X, p_\lambda]$ and $[Z, q]$ are not adjusted, but our defining relations do not fix these positions. So, we use these defining relations to represent the equation by the following equation between words

$$u_\lambda[X, p_\lambda]p_\lambda^{\ell''}r_\lambda v_\lambda w'u_\rho p_\rho^{r'}r_\rho[Y, p_\rho](g, Y) = u_\lambda[Z, q]p_\lambda^{\ell'''}r_\lambda Z. \tag{3}$$

The morphism σ solves this equation. Moreover, in our concrete situation we have $v_\lambda w'u_\rho = vu_\nu p_\nu^\nu r_\nu u$; and again, we content ourselves to consider the special case where $u_\nu p_\nu^\nu r_\nu$ is the only occurrence of very long δ-periodic factor in $v_\lambda w'u_\rho$. Ignoring u_λ on the left, the remaining task is to compress the equation (where $v_\lambda \le vu_\nu$ and $\overline{u_\rho} \le \overline{u}\,\overline{v_\nu}$)

$$[X, p_\lambda]p_\lambda^{\ell''}r_\lambda vu_\nu p_\nu^\nu r_\nu v_\nu up_\rho^{r'}r_\rho[Y, p_\rho](g, Y) = [Z, q]p_\lambda^{\ell'''}r_\lambda Z \tag{4}$$

with respect to the solution σ. The crucial idea comes next: we use a larger alphabet of constants, we change the type of variables and we introduce more defining relations. For each $\lambda \in \Lambda$ we introduce a new constant, denoted as $[p_\lambda, r_\lambda, \lambda]$, and for each variable $[V, p]$ we introduce a constant $[p]$. Thus, $[p_\lambda, r_\lambda, \lambda]$ and $[p]$ are fresh letters. We also let act G on these letters in the obvious way, so we actually introduce more letters. Let h be the morphism defined by $h([p_\lambda, r_\lambda, \lambda]) = p_\lambda r_\lambda$ and $h([p]) = p$, it means h compresses the words $p_\lambda r_\lambda$ and p into single letters, then Equation (4) is the image under h of the equation

$$[X, p_\lambda][p_\lambda]^{\ell''-1}\,[p_\lambda, r_\lambda, \lambda]\,vu_\nu[p_\nu]^{\nu-1}\,[p_\nu, r_\nu, \nu]\,v_\nu u[p_\rho]^{r'-1}\,[p_\rho, r_\rho, \rho]\,[Y, p_\rho](g, Y) \tag{5}$$

$$= [Z, q][p_\lambda]^{\ell'''-1}\,[p_\lambda, r_\lambda, \lambda]\,Z \tag{6}$$

To have a visual notation we color the letters of the form $[p_\lambda, r_\lambda, \lambda]$ *green*. The procedure continues by redefining the type of a twisted variable $[V, p]$ as the letter $[p]$. We augment θ by more defining relations:

$$\{[V, p][p] = [p][V, p] \mid [V, p] \text{ twisted variable}\} \cup \left\{ [p_\lambda, r_\lambda, \lambda]\,[sr_\lambda] = [p_\lambda]\,[p_\lambda, r_\lambda, \lambda] \;\middle|\; p_\lambda = r_\lambda s \right\}.$$

It is not hard to see that we find a solution σ' of the new equation over the larger alphabet of constants such that $h\sigma' = \sigma h$ which is needed to prove the EDT0L property. The

remaining procedure is essentially the same as in [3]: using transformations either based on substitutions $[V,p] \mapsto [V,p][p]$ and $[V,p] \mapsto 1$ or homomorphisms based on $[p] \mapsto [p][p]$ and $[p_\lambda, r_\lambda, \lambda] \mapsto [p_\lambda, r_\lambda, \lambda]\,[p_\lambda]$ we can compress the above equation and simultaneously the solution such that the equation becomes its final form. We finish δ-periodic compression with

$$[p_\lambda, r_\lambda, \lambda]\, v u_\nu\, [p_\nu, r_\nu, \nu]\, v_\nu u\, [p_\rho, r_\rho, \rho]\, (g, Y) = [Z, q]\, [p_\lambda, r_\lambda, \lambda]\, Z. \tag{7}$$

The typed variables are gone, the letters $[p]$ are not visible anymore, moreover, the new solution doesn't use them. We are back in a free monoid, because none of the defining relations is used anymore. Note that Equation (7) is shorter than the original equation. Indeed, while the initial increase in the length of the equation is in $\mathcal{O}(n\delta)$, each green letter represents the inner part of a very long δ-periodic word of length at least 6δ.

▶ **Proposition 5.** *Let $E_s = (W_s, B_s, \mathcal{X}_s, \emptyset, \mu_s)$ be the state where we started δ-periodic-compression with $|W_s| \geq 8\delta n$; and let $E_t = (W_t, B_t, \mathcal{X}_t, \emptyset, \mu_t)$ the standard state where we finish δ-periodic-compression, and $(W, B, \mathcal{X}, \theta, \mu)$ any state which we have seen on the path from E_s to E_t during the procedure. Then we have $|W_t| \leq |W_s| + 20\delta n$ and $|W| \leq |W_s| + \mathcal{O}(\delta n)$. Moreover, let $n_{\text{new}} = \sum_{b \in B_t \setminus B_s} |W_t|_b$. If $n_{\text{new}} \geq 10n$, then $|W_t| < |W_s|$.*

▶ Remark. Note that n_{new} is the number of green letters we see in W_t. Let σ be the solution after δ-periodic-compression, then for $X \in \mathcal{X}_t$ the length of a δ-periodic prefix (and suffix resp.) is bounded by 3δ. Hence, there is no very long δ-periodic prefix or suffix in $\sigma(X)$.

3.4 Twisted pair-compression

We place ourselves after a sequence of rounds of popping out letters for each variable, alphabet-reduction, and δ-periodic compression. We are at a standard state $E = (W, B, \mathcal{X}, \emptyset, \mu)$ where $\emptyset \neq \mathcal{X} \subseteq \mathcal{V}$. Without restriction, we may assume that $|W| \in \Theta(|G|n^2)$ and that the number of visible green letters is at most $10n$: our construction ensures that $|W| \in \mathcal{O}(|G|n^2)$, and we can always pop out letters to make the equation longer; and if the number of visible green letters exceeds $10n$ then according to Proposition 5 the most recent δ-periodic-compression had decreased the length of the equation, so we can perform another round.

Throughout, it is possible to write $W = U_1 \# u(f,X) w(g,Y) v \# U_2 \overline{U_2'} \,\overline{\#}\, \overline{v} \overline{Z} \overline{u} \,\overline{\#}\, \overline{U_1'}$ with $|U_i'|_\# = |U_i'|_{\overline{\#}}$, $i = 1, 2$. Here $u(f,X)w(g,Y)v = uZv$ is called a *local equation*. For simplicity we may assume that $u, v, w \in B^*$ and that $(f,X), (g,Y), Z = (1,Z)$ are twisted variables. Moreover, we may assume that for each local equation its "dual" equation $\overline{v}(g,\overline{Y})\overline{w}(f,\overline{X})\overline{u} = \overline{v}\overline{Z}\overline{v}$ is also part of the system encoded in W. Since W is long, we can assume that $|uwv|$ is long, too. Since there are at most $\mathcal{O}(n)$ green letters, there are long intervals without green letters. The goal is to compress enough pairs $ab \leq W$ of constants into single letters without causing any conflict or overlap with other pairs or variables that are connected via twisting. We compress pairs according to an equivalence relation between positions. The idea is that whenever we modify a solution at position i, then we must modify $\sigma(W)$ at all equivalent positions $j \equiv i$.

The notion of *equivalent positions* is defined for a given solution σ, it has a reasonably intuitive definition. We write $W = U\overline{V}$ with $\sigma(U) = \sigma(V)$ and $\sigma(U) = a_1 \cdots a_m$ with $a_i \in B$. We associate with U (resp. V) the interval $[1, m] \subseteq \mathbb{N}$ (resp. $[m+1, 2m]$) of positions and we let $i \sim m + i$ for $1 \leq i \leq m$. We say that position i sits directly "above" $m + i$, see Figure 2.

Each occurrence of a twisted variable (f, X) in UV corresponds to some interval of length $|\sigma(X)|$ in $[1, 2m]$ and we identify the i-th positions in each of these intervals for $1 \leq i \leq |\sigma(X)|$. Identified positions are represented by a unique position corresponding to

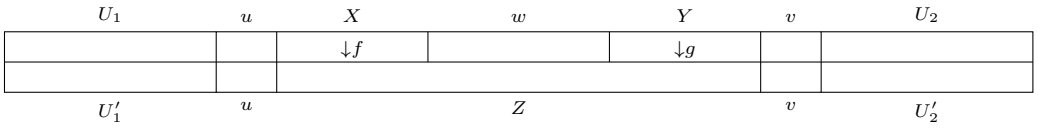


Figure 2 $W = U\overline{V}$ viewed as U on top and V on the bottom and $\sigma(U) = \sigma(V)$ in the middle.

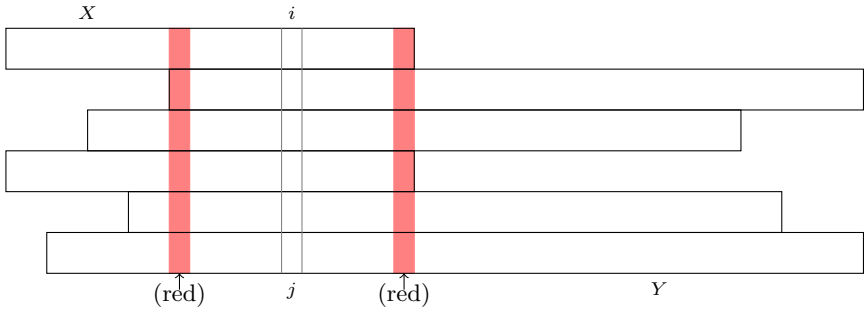


Figure 3 Red positions. We use $\sim$ to put $i \approx j$ into a "domino tower".

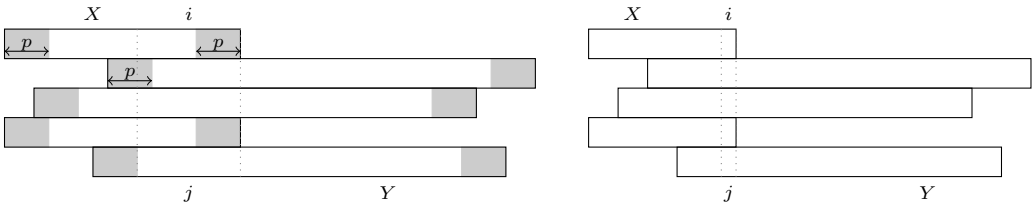
the leftmost occurrence of a twisted variable (f, X) in U. This interval is denoted by $I(X)$. Thus, we identify various positions and we carry over the relation $\sim$: if i and j are identified with i' and j' and if $i' \sim j'$, then we also let $i \sim j$. By $\approx$ we denote the generated equivalence relation of $\sim$. The relation $\approx$ can be visualized in so-called *domino towers* as in Figure 3. Clearly, we may have $i \approx j$ for various $i, j \in I(X)$. For example, an equation $(f, X)a = bX$ forces $i \approx j$ for all $i, j \in I(X)$. There is also a natural notion of duality: $I(X)$ and $I(\overline{X})$ are disjoint, but if we change $\sigma(X)$ at the first position, we must change $\sigma(\overline{X})$ at the last position. Thus, for the i-th position in $I(X)$ we let $\overline{i}$ be the $(|\sigma(X)| - i + 1)$-st position in $I(\overline{X})$; and we write $i \leftrightarrow \overline{i}$. Finally, we let $\equiv \subseteq [1, 2m] \times [1, 2m]$ be the equivalence relation generated by $\approx$ and $\leftrightarrow$. Clearly, if $i \approx j \leftrightarrow \overline{j}$ and the label at position i is $a \in B$, then a labels j and $\overline{a}$ labels $\overline{j}$.

Positions at the borders of some $\sigma(X)$ inside $\sigma(W)$ play a special role because we cannot compress over borders. We color the first and last position in each $I(X)$ *red* (unless it has already the color green) to signal "danger". We color red all positions equivalent to a red position, too. Since the set of green positions is closed under equivalence (they are the fresh letters $[p_\lambda, r_\lambda, \lambda]$), no conflict between red and green is introduced here. It follows that there are at most n pairwise different equivalence classes of red positions.

We extend the notion of equivalence to intervals (without red positions). Let $p \in \mathbb{N}$. We directly link an interval $[i, i + p]$ of positions in $\sigma(X)$ (resp. w, $\sigma(Y)$) to $[j, j + p]$ in $\sigma(Z)$ if there is an equation, for example like $u(f, X)w(g, Y)v = uZv$, such that $\sigma(X)[i, i + p]$ (resp. $w[i, i + p]$, $\sigma(Y)[i, i + p]$) sits directly above the $\sigma(Z)[j, j + p]$; and we write $[i, i + p] \sim [j, j + p]$ in this case. For each interval $[i, i + p]$ of positions in $\sigma(X)$ we also let $[i, i + p] \leftrightarrow [\overline{i + p}, \overline{i}]$. As above, we let $\approx$ and $\equiv$ be the generated equivalence relations of $\sim$ resp. $\sim \cup \leftrightarrow$. Since $i \sim j \iff \overline{i} \sim \overline{j}$ we can deduce

$$[i, i + p] \equiv [j, j + p] \iff [i, i + p] \approx [j, j + p] \vee [i, i + p] \approx [\overline{j + p}, \overline{j}]. \tag{8}$$

▶ **Lemma 6.** *Let $[i - 1, i, i + 1, i + 2]$ be an interval without any red position and where the four positions are pairwise inequivalent. Consider $[i, i + 1] \equiv [j, j + 1] \equiv [k, k + 1]$. Then either $k = j$ and $[j, j + 1] \not\approx [\overline{k} - 1, \overline{k}]$ or $[j, j + 1] \cap [k, k + 1] = \emptyset$.*



(a) $W : [i-p, i+p] \sim [j-p, j+p]$ **(b)** $W_p : i \sim j$

Figure 4 Example illustrating the proof of Lemma 7.

Let $p \in \mathbb{N}$ and σ be a solution for W. For each X we do:

- if $|\sigma(X)| \leq 2p$, then replace X by $\sigma(X)$ and remove X from the set of variables;
- if $|\sigma(X)| > 2p$, then write $\sigma(X) = uwv$ with $|u| = |v| = p$ and replace X by uXv. Change the interval $I(X) = [l, r]$ to $I_p(X) = [l + p, r - p]$. (So, it is smaller.)

Denote the new solution for W_p defined by that procedure by σ_p.

▶ **Lemma 7.** *Let i and j be positions in $\sigma_p(W_p) = \sigma(W)$ which belong to variables in W_p. This means $i, j \in \bigcup\{I_p(X) \mid X \in \mathcal{X}_p\}$. Then we have $i \sim j$ (resp. $i \leftrightarrow j$) for W_p and σ_p if and only if $[i-p, i+p] \sim [j-p, j+p]$ (resp. $[i-p, i+p] \leftrightarrow [j-p, j+p]$) for W and σ.*

We define and fix $\delta = |G|\,\varepsilon$ and $\varepsilon = 30n$. We start at a standard state $E = (W, B, \mathcal{X}, \emptyset, \mu)$ together with a solution σ. For simplicity, we assume that all local equations have the form $u(f, X)w(g, Y)v = uZv$. Moreover, when we start pair-compression (directly after δ-periodic-compression) there are some green letters and corresponding green visible positions.

1. For every X in some order do: either replace X by $\sigma(X)$ (if $|\sigma(X)| \leq 10\delta$) or write $\sigma(X) = ux$ with $|u| = 10\delta$; replace X by $\tau(X) = uX$; rename the new equation and new solution as (E, σ). Define the intervals $I(X)$ as done above color *red* positions in $\sigma(W)$ which are equivalent to a first or last position in $I(X)$ unless they are green.
2. **while** there is an interval $[i-1, i, i+1, i+2]$ such that (1) all four positions are pairwise inequivalent, (2) no position is colored, and (3) all positions are visible **do**

   a. Let ab the label of the middle interval $[i, i+1]$. Choose fresh letter c and define a morphism h by $h(c) = ab$. (Hence, $f(c) = c \iff f(ab) = ab$, too.) Whenever $[i, i+1] \approx [j, j+1]$, then the label of $[j, j+1]$ is $f(ab)$ for some $f \in G$. Replace each of the intervals $[j, j+1]$ and $[\overline{j}-1, \overline{j}]$ by a single new position and label this position with $f(c)$ and $f(\overline{c})$ resp. There is no conflict in this relabeling by Lemma 6. Since there is no red position, there is no "crossing" of the intervals $[j, j+1]$ or $[\overline{j}-1, \overline{j}]$. So, this gives a new but shorter equation W'. We have $h(W') = W$ and new solution σ' such that $h\sigma'(W') = \sigma(W)$ There is a new numbering for the positions, but the colored positions can still be identified.

   b. Define $B' = B \cup \{f(c), f(\overline{c}) \mid f \in G\}$ and $E' = (W', B', \mathcal{X}, \emptyset, \mu')$.

   c. Rename (E', σ') as (E, σ) and transfer the induced coloring.

   end while

If we started the procedure with W and the while loop with W_ℓ, then the loop terminates with an equation W' and we introduced at most $|G|\,(|W_\ell| - |W'|)$ new letters. It is also clear that $|W'| \leq |W| + 20\delta n$ since any increase of length is due to the first steps, where we replaced each variable X either by $\sigma(X)$ or by uXv. The worst case for $|W'|$ is that no compression took place. However, we assume that there at most $10n$ green letters. Hence, we can use the following fact.

▶ **Proposition 8.** *Let (E, σ) with equation W just after a δ-periodic-compression where at most $10n$ green letters are visible. If $|W| \in 20\ell\delta n + \mathcal{O}(\delta n)$, then the pair-compression procedure outputs an equation W' such that $|W'| \leq |W| + 20\delta n$ and $|W'| \leq \frac{59|W|}{60} + \mathcal{O}(\delta n)$.*

Let us highlight that Proposition 8 is the key step in the proof of Theorem 1 and it is here where twisted conjugacy comes into play. Following any given solution at the initial state, it bounds the lengths of all intermediate equations in $\mathcal{O}(\delta n) = \mathcal{O}(|G|n^2)$. Since at a standard state we can perform an alphabet reduction we can bound the size of the extended alphabet C in $\mathcal{O}(|G|^2 n^2)$. Moreover, the number of untyped variables is never increasing. Typed variables disappear and reappear, but their number never grows beyond the size of C.

After δ-periodic compression, no $\sigma(X)$ started or ended in a very long δ-periodic word. In the procedure above either X vanished or we replaced X by uXv where $|u| = |v| = 10\delta$. We carefully colored some position red after that replacement. Consider the new equation with the new solution just after that step; and rename the corresponding pair as (W, σ). Consider positions $i < k$ in $\sigma(W)$ such that no position k with $i \leq k \leq j$ is green. With the help of Proposition 3, Lemma 7 and "domino towers" as depicted in Figure 4, one can show the following fact: if $i \equiv k \equiv j$ for some k, then $|j - i| > \varepsilon$. The fact is not obvious but extremely useful: knowing that equivalent positions are far apart allows one to find enough intervals of length four, such that pair-compression reduces their length to at most three by Lemma 6.

Putting all this together, the overall compression method has the following high-level description. Start at the initial state E_{init} with a given initial entire solution $(\text{id}_{A^*}, \sigma_{\text{init}})$.

begin compression
Rename E_{init} as $E = (W, B, \mathcal{X}, \emptyset, \mu)$; rename $(\text{id}_{A^*}, \sigma_{\text{init}})$ as (α, σ).
Repeat the following loop until $\mathcal{X} = \emptyset$.
begin loop
1. Pop out letters from variables until $|W| \geq 100\delta n$.
2. Define $\kappa > 0$ by $\kappa\delta n = |W|$. Call δ-periodic-compression (starting with an alphabet-reduction), and let W' denote the equation at the end of the procedure.
3. If $|W'| < \kappa\delta n$, then do nothing, else call pair-compression.
end loop
end compression

Proposition 8 implies that $\kappa \in \mathbb{Q}$ is bounded above by some effective constant in $\mathcal{O}(1)$. Defining a weight in $\mathbb{N}^4$ (ordered lexicographically) by

$$\|E, \alpha, \sigma\| = \left(\sum_{X \text{ has no type}} |\alpha\sigma(X)|, \sum_{X \text{ is typed}} |\alpha\sigma(X)|, |W|, |B| \right)$$

finally shows that the compression method terminates for every given solution because every step in the procedures is weight-reducing. This means our algorithm finds all solutions. This finishes the outline of the proof of Theorem 1.

Acknowledgements. We thank the anonymous referees for very helpful feedback.

References

1 Peter R. J. Asveld. Controlled iteration grammars and full hyper-AFL's. *Information and Control*, 34(3):248–269, 1977. doi:10.1016/S0019-9958(77)90308-4.

2 Gilbert Baumslag, Alexei Myasnikov, and Vladimir Remeslennikov. Algebraic geometry over groups. In *Algorithmic problems in groups and semigroups (Lincoln, NE, 1998)*, Trends Math., pages 35–50. Birkhäuser Boston, Boston, MA, 2000. doi:10.1007/978-1-4612-1388-8_3.

3 Laura Ciobanu, Volker Diekert, and Murray Elder. Solution sets for equations over free groups are EDT0L languages. *Internat. J. Algebra Comput.*, 26(5):843–886, 2016. Conference version in ICALP 2015, LNCS 9135. doi:10.1142/S0218196716500363.

4 François Dahmani and Vincent Guirardel. Foliations for solving equations in groups: free, virtually free, and hyperbolic groups. *J. Topol.*, 3(2):343–404, 2010. doi:10.1112/jtopol/jtq010.

5 Volker Diekert and Murray Elder. Solutions of twisted word equations, EDT0L languages, and context-free groups. *ArXiv e-prints*, January 2017. URL: https://arxiv.org/abs/1701.03297, arXiv:1701.03297.

6 Volker Diekert, Artur Jeż, and Wojciech Plandowski. Finding all solutions of equations in free groups and monoids with involution. *Inform. and Comput.*, 251:263–286, 2016. Conference version in Proc. CSR 2014, LNCS 8476 (2014). doi:10.1016/j.ic.2016.09.009.

7 Volker Diekert and Armin Weiß. Context-free groups and Bass-Serre theory. *ArXiv e-prints*, July 2013. arXiv:1307.8297.

8 Martin J. Dunwoody. The accessibility of finitely presented groups. *Inventiones Mathematicae*, 81(3):449–457, 1985. doi:10.1007/BF01388581.

9 Andrzej Ehrenfeucht and Grzegorz Rozenberg. On some context free languages that are not deterministic ET0L languages. *RAIRO Theor. Inform. Appl.*, 11:273–291, 1977.

10 Samuel Eilenberg. *Automata, languages, and machines. Vol. A*. Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York, 1974. Pure and Applied Mathematics, Vol. 58.

11 Julien Ferté, Nathalie Marin, and Géraud Sénizergues. Word-mappings of level 2. *Theory Comput. Syst.*, 54:111–148, 2014. doi:10.1007/s00224-013-9489-5.

12 Robert H. Gilman. Personal communication, 2012.

13 Robert H. Gilman, Susan Hermiller, Derek F. Holt, and Sarah Rees. A characterisation of virtually free groups. *Arch. Math. (Basel)*, 89(4):289–295, 2007. doi:10.1007/s00013-007-2206-3.

14 Mikhael Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987. doi:10.1007/978-1-4613-9586-7_3.

15 Sanjay Jain, Alexei Miasnikov, and Frank Stephan. The complexity of verbal languages over groups. In *Proceedings of the 2012 27th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 405–414. IEEE Computer Soc., Los Alamitos, CA, 2012. doi:10.1109/LICS.2012.50.

16 Artur Jeż. Recompression: a simple and powerful technique for word equations. *J. ACM*, 63(1):Art. 4, 51, 2016. Conference version in Proc. STACS 2013. doi:10.1145/2743014.

17 Artur Jeż. Word Equations in Nondeterministic Linear Space, 2017. doi:10.4230/LIPIcs.ICALP.2017.95.

18 Abe Karrass, Alfred Pietrowski, and Donald Solitar. Finite and infinite cyclic extensions of free groups. *J. Austral. Math. Soc.*, 16:458–466, 1973. Collection of articles dedicated to the memory of Hanna Neumann, IV. doi:10.1017/S1446788700015445.

19 Dietrich Kuske and Markus Lohrey. Logical aspects of Cayley-graphs: the group case. *Ann. Pure Appl. Logic*, 131(1-3):263–286, 2005. doi:10.1016/j.apal.2004.06.002.

20 Markus Lohrey and Géraud Sénizergues. Theories of HNN-extensions and amalgamated products. In *Automata, languages and programming. Part II*, volume 4052 of *Lecture Notes in Comput. Sci.*, pages 504–515. Springer, Berlin, 2006. doi:10.1007/11787006_43.

21 Gennadiĭ S. Makanin. The problem of solvability of equations in a free semigroup. *Math. Sbornik*, 103:147–236, 1977. English transl. in Math. USSR Sbornik 32 (1977).

22 Victor Mazurov and Evgeny Khukhro. Unsolved Problems in Group Theory. The Kourovka Notebook. No. 18 (English version). *ArXiv e-prints*, January 2014. arXiv:1401.0300.

23 David E. Muller and Paul E. Schupp. Groups, the theory of ends, and context-free languages. *J. Comput. System Sci.*, 26(3):295–310, 1983. doi:10.1016/0022-0000(83)90003-X.

24 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *Journal of the ACM*, 51:483–496, 2004. Conference version in FOCS'99. doi:doi:10.1145/990308.990312.

25 Wojciech Plandowski and Wojciech Rytter. Application of Lempel-Ziv encodings to the solution of word equations. In K. G. Larsen et al., editors, *Proc. 25th International Colloquium Automata, Languages and Programming (ICALP'98), Aalborg (Denmark), 1998*, volume 1443 of *Lecture Notes in Computer Science*, pages 731–742, Heidelberg, 1998. Springer-Verlag.

26 Eliyahu Rips and Zlil Sela. Canonical representatives and equations in hyperbolic groups. *Invent. Math.*, 120(3):489–512, 1995. doi:10.1007/BF01241140.

27 Grzegorz Rozenberg and Arto Salomaa. *The Book of L.* Springer, 1986.

28 Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of formal languages. Vol. 1.* Springer-Verlag, Berlin, 1997. Word, language, grammar. doi:10.1007/978-3-642-59126-6.

29 Géraud Sénizergues. An effective version of Stallings' theorem in the case of context-free groups. In *Automata, languages and programming (Lund, 1993)*, volume 700 of *Lecture Notes in Comput. Sci.*, pages 478–495. Springer, Berlin, 1993. doi:10.1007/3-540-56939-1_96.

30 Géraud Sénizergues. On the finite subgroups of a context-free group. In *Geometric and computational perspectives on infinite groups (Minneapolis, MN and New Brunswick, NJ, 1994)*, volume 25 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 201–212. Amer. Math. Soc., Providence, RI, 1996.

31 Jean-Pierre Serre. *Trees.* Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2003. Translated from the French original by John Stillwell, Corrected 2nd printing of the 1980 English translation.

Pumping Lemma for Higher-Order Languages*†

Kazuyuki Asada[1] and Naoki Kobayashi[2]

1 **The University of Tokyo, Tokyo, Japan**
asada@kb.is.s.u-tokyo.ac.jp
2 **The University of Tokyo, Tokyo, Japan**
koba@is.s.u-tokyo.ac.jp

Abstract

We study a pumping lemma for the word/tree languages generated by higher-order grammars. Pumping lemmas are known up to order-2 word languages (i.e., for regular/context-free/indexed languages), and have been used to show that a given language does not belong to the classes of regular/context-free/indexed languages. We prove a pumping lemma for word/tree languages of arbitrary orders, modulo a conjecture that a higher-order version of Kruskal's tree theorem holds. We also show that the conjecture indeed holds for the order-2 case, which yields a pumping lemma for order-2 tree languages and order-3 word languages.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases pumping lemma, higher-order grammars, Kruskal's tree theorem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.97

1 Introduction

We study a pumping lemma for higher-order languages, i.e., the languages generated by higher-order word/tree grammars where non-terminals can take higher-order functions as parameters. The classes of higher-order languages [26, 18, 4, 5, 6] form an infinite hierarchy, where the classes of order-0, order-1, and order-2 languages are those of regular, context-free and indexed languages. Higher-order grammars and languages have been extensively studied by Damm [4] and Engelfriet [5, 6] and recently re-investigated in the context of model checking and program verification [9, 20, 15, 24, 11, 16, 12, 23].

Pumping lemmas [2, 7] are known up to order-2 word languages, and have been used to show that a given language does not belong to the classes of regular/context-free/indexed languages. To our knowledge, however, little is known about languages of order-3 or higher. Pumping lemmas [21, 12] are also known for higher-order *deterministic* grammars (as generators of infinite *trees*, rather than tree languages), but they cannot be applied to non-deterministic grammars.

In the present paper, we state and prove a pumping lemma for unsafe[1] languages of arbitrary orders modulo an assumption that a "higher-order version" of Kruskal's tree theorem [17, 19] holds. Let $\preceq$ be the homeomorphic embedding on finite ranked trees[2], and

* A full version of the paper is available at http://arxiv.org/abs/1705.10699.

† This work was supported by JSPS Kakenhi 15H05706.

[1] See, e.g., [16] for the distinction between safe vs unsafe languages; the class of unsafe languages subsumes that of safe languages.

[2] I.e., $T_1 \preceq T_2$ if there exists an injective map from the nodes of T_1 to those of T_2 that preserves the labels of nodes and the ancestor/descendant-relation of nodes; see Section 2 for the precise definition.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 97; pp. 97:1–97:14

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

$\prec$ be the strict version of $\preceq$. The statement of our pumping lemma[3] is that for any order-n infinite tree language L, there exist a constant c and a strictly increasing infinite sequence of trees $T_0 \prec T_1 \prec T_2 \prec \cdots$ in L such that $|T_i| \leq \mathbf{exp}_n(ci)$ for every $i \geq 0$, where $\mathbf{exp}_0(x) = x$ and $\mathbf{exp}_{n+1}(x) = 2^{\mathbf{exp}_n(x)}$. Due to the correspondence between word/tree languages [4, 1], it also implies that for any order-n infinite word language L (where $n \geq 1$), there exist a constant c and a strictly increasing infinite sequence of words $w_0 \prec w_1 \prec w_2 \prec \cdots$ in L such that $|w_i| \leq \mathbf{exp}_{n-1}(ci)$ for every $i \geq 0$, where $\prec$ is the subsequence relation. The pumping lemma can be used, for example, to show (modulo the conjecture) that the order-$(n+1)$ language $\{a^{\mathbf{exp}_n(k)} \mid k \geq 0\}$ does not belong to the class of order-n word languages, for $n > 0$. Thus the lemma would also provide an alternative proof of the strictness of the hierarchy of the classes of higher-order languages.[4]

We now informally explain the assumption of "higher-order Kruskal's tree theorem" (see Section 2 for details). Kruskal's tree theorem [17, 19] states that the homeomorphic embedding $\preceq$ is a well-quasi order, i.e., that for any infinite sequence of trees $T_0, T_1, T_2, \ldots$, there exist $i < j$ such that $T_i \preceq T_j$. The homeomorphic embedding $\preceq$ can be naturally lifted (e.g. via the logical relation) to a family of relations $(\preceq_\kappa)_\kappa$ on higher-order tree functions of type κ. Our conjecture of "higher-order Kruskal's theorem" states that, for every simple type κ, $\preceq_\kappa$ is also a well-quasi order on the functions expressed by the simply-typed λ-terms. We prove that the conjecture indeed holds up to order-2 functions, if we take $\preceq_\kappa$ as the logical relation induced from the homeomorphic embedding $\preceq$. Thus, our pumping "lemma" is indeed true for order-2 tree languages and order-3 word languages. To our knowledge, the pumping lemma for those languages is novel. The conjecture remains open for order-3 or higher, which should be of independent interest.

Our proof of the pumping lemma (modulo the conjecture) uses the recent work of Parys [23] on an intersection type system for deciding the infiniteness of the language generated by a given higher-order grammar, and our previous work on the relationship between higher-order word/tree languages [1].

The rest of this paper is organized as follows. Section 2 prepares several definitions and states our pumping lemma and the conjecture more formally. Section 3 derives some corollaries of Parys' result [23]. Section 4 prepares a simplified and specialized version of our previous result [1]. Using the results in Sections 3 and 4, we prove our pumping lemma (modulo the conjecture) in Section 5. Section 6 proves the conjecture on higher-order Kruskal's tree theorem for the order-2 case, by which we obtain the (unconditional) pumping lemma for order-2 tree languages and order-3 word languages. Section 7 discusses related work and Section 8 concludes.

2 Preliminaries

We first give basic definitions needed for explaining our main theorem. We then state the main theorem and provide an overview of its proof.

[3] This should perhaps be called a pumping "conjecture" since it relies on the conjecture of the higher-order Kruskal's tree theorem.

[4] The strictness of the hierarchy of higher-order *safe* languages has been shown by Engelfriet [5] using a complexity argument, and Kartzow [8] observed that essentially the same argument is applicable to obtain the strictness of the hierarchy of unsafe languages as well. Their argument cannot be used for showing that a particular language does not belong to the class of order-n languages.

2.1 λ-terms and Higher-order Grammars

This section gives basic definitions for terms and higher-order grammars.

▶ **Definition 1** (Types and Terms). The set of *simple types*, ranged over by κ, is given by: $\kappa ::= \mathtt{o} \mid \kappa_1 \to \kappa_2$. The order of a simple type κ, written $\mathtt{order}(\kappa)$ is defined by $\mathtt{order}(\mathtt{o}) = 0$ and $\mathtt{order}(\kappa_1 \to \kappa_2) = \max(\mathtt{order}(\kappa_1) + 1, \mathtt{order}(\kappa_2))$. The type $\mathtt{o}$ describes trees, and $\kappa_1 \to \kappa_2$ describes functions from κ_1 to κ_2. The set of $\lambda^{\to,+}$-*terms* (or *terms*), ranged over by s, t, u, v, is defined by:

$$t ::= x \mid a\, t_1 \cdots t_k \mid t_1\, t_2 \mid \lambda x : \kappa.t \mid t_1 + t_2\,.$$

Here, x ranges over variables, and a over constants (which represent tree constructors). Variables are also called *non-terminals*, ranged over by x, y, z, f, g, A, B; and constants are also called *terminals*. A ranked alphabet Σ is a map from a finite set of terminals to natural numbers called *arities*; we implicitly assume a ranked alphabet whose domain contains all terminals discussed, unless explicitly described. $+$ is non-deterministic choice. As seen below, our simple type system forces that a terminal must be fully applied; this does not restrict the expressive power, as $\lambda x_1, \ldots, x_k.a\, x_1 \cdots x_k$ is available. We often omit the type κ of $\lambda x : \kappa.t$. A term is called an *applicative term* if it does not contain λ-abstractions nor $+$, and called a $\lambda^{\to}$-*term* if it does not contain $+$. As usual, we identify terms up to the α-equivalence, and implicitly apply α-conversions.

A (simple) type environment $\mathcal{K}$ is a sequence of type bindings of the form $x : \kappa$ such that if $\mathcal{K}$ contains $x : \kappa$ and $x' : \kappa'$ in different positions then $x \neq x'$. In type environments, non-terminals are also treated as variables. A term t has type κ under $\mathcal{K}$ if $\mathcal{K} \vdash_{\mathtt{ST}} t : \kappa$ is derivable from the following typing rules.

$$\frac{}{\mathcal{K}, x : \kappa, \mathcal{K}' \vdash_{\mathtt{ST}} x : \kappa} \qquad \frac{\Sigma(a) = k \qquad \mathcal{K} \vdash_{\mathtt{ST}} t_i : \mathtt{o} \ \ (\text{for each } i \in \{1, \ldots, k\})}{\mathcal{K} \vdash_{\mathtt{ST}} a\, t_1 \cdots t_k : \mathtt{o}}$$

$$\frac{\mathcal{K} \vdash_{\mathtt{ST}} t_1 : \kappa_2 \to \kappa \qquad \mathcal{K} \vdash_{\mathtt{ST}} t_2 : \kappa_2}{\mathcal{K} \vdash_{\mathtt{ST}} t_1\, t_2 : \kappa} \qquad \frac{\mathcal{K}, x : \kappa_1 \vdash_{\mathtt{ST}} t : \kappa_2}{\mathcal{K} \vdash_{\mathtt{ST}} \lambda x : \kappa_1.t : \kappa_1 \to \kappa_2} \qquad \frac{\mathcal{K} \vdash_{\mathtt{ST}} t_1 : \mathtt{o} \qquad \mathcal{K} \vdash_{\mathtt{ST}} t_2 : \mathtt{o}}{\mathcal{K} \vdash_{\mathtt{ST}} t_1 + t_2 : \mathtt{o}}$$

We consider below only well-typed terms. Note that given $\mathcal{K}$ and t, there exists at most one type κ such that $\mathcal{K} \vdash_{\mathtt{ST}} t : \kappa$. We call κ the type of t (with respect to $\mathcal{K}$). We often omit "with respect to $\mathcal{K}$" if $\mathcal{K}$ is clear from context. The (internal) *order* of t, written $\mathtt{order}_{\mathcal{K}}(t)$, is the largest order of the types of subterms of t, and the *external order* of t, written $\mathtt{eorder}_{\mathcal{K}}(t)$, is the order of the type of t (both with respect to $\mathcal{K}$). We often omit $\mathcal{K}$ when it is clear from context. For example, for $t = (\lambda x : \mathtt{o}.x)\mathtt{e}$, $\mathtt{order}_{\emptyset}(t) = 1$ and $\mathtt{eorder}_{\emptyset}(t) = 0$.

We call a term t *ground* (with respect to $\mathcal{K}$) if $\mathcal{K} \vdash_{\mathtt{ST}} t : \mathtt{o}$. We call t a (finite, Σ-ranked) *tree* if t is a closed ground applicative term consisting of only terminals. We write $\mathbf{Tree}_{\Sigma}$ for the set of Σ-ranked trees, and use the meta-variable π for trees.

The set of *contexts*, ranged over by C, D, G, H, is defined by $C ::= [\,] \mid C\, t \mid t\, C \mid \lambda x.C$. We write $C[t]$ for the term obtained from C by replacing $[\,]$ with t. Note that the replacement may capture variables; e.g., $(\lambda x.[\,])[x]$ is $\lambda x.x$. We call C a $(\mathcal{K}', \kappa')$-$(\mathcal{K}, \kappa)$-*context* if $\mathcal{K} \vdash_{\mathtt{ST}} C : \kappa$ is derived by using axiom $\mathcal{K}' \vdash_{\mathtt{ST}} [\,] : \kappa'$. We also call a $(\emptyset, \kappa')$-$(\emptyset, \kappa)$-context a κ'-κ-*context*. The (internal) *order* of a $(\mathcal{K}', \kappa')$-$(\mathcal{K}, \kappa)$-context, is the largest order of the types occurring in the derivation of $\mathcal{K} \vdash_{\mathtt{ST}} C : \kappa$. A context is called a $\lambda^{\to}$-*context* if it does not contain $+$.

We define the *size* $|t|$ of a term t by: $|x| := 1$, $|a\, t_1 \cdots, t_k| := 1 + |t_1| + \cdots + |t_k|$, $|s\, t| := |s| + |t| + 1$, $|\lambda x.t| := |t| + 1$, and $|s + t| := |s| + |t| + 1$. The size $|C|$ of a context C is defined similarly, with $|[\,]| := 0$.

▶ **Definition 2** (Reduction and Language)**.** The set of *(call-by-name) evaluation contexts* is defined by:

$$E ::= [\,]\,t_1 \cdots t_k \mid a\,\pi_1 \cdots \pi_i\,E\,t_1 \cdots t_k$$

and the *call-by-name reduction* for (possibly open) ground terms is defined by:

$$E[(\lambda x.t)t'] \longrightarrow E[[t'/x]t] \qquad E[t_1 + t_2] \longrightarrow E[t_i] \quad (i = 1, 2)$$

where $[t'/x]t$ is the usual capture-avoiding substitution. We write $\longrightarrow^*$ for the reflexive transitive closure of $\longrightarrow$. A *call-by-name normal form* is a ground term t such that $t \not\longrightarrow t'$ for any t'. For a closed ground term t, we define the *tree language* $\mathcal{L}(t)$ *generated by* t by $\mathcal{L}(t) := \{\pi \mid t \longrightarrow^* \pi\}$. For a closed ground $\lambda^{\to}$-term t, $\mathcal{L}(t)$ is a singleton set $\{\pi\}$; we write $\mathcal{T}(t)$ for such π and call it *the tree of* t.

Note that $t \longrightarrow^* t'$ implies $[s/x]t \longrightarrow^* [s/x]t'$, and that the set of call-by-name normal forms equals the set of trees and ground terms of the form $E[x]$.

For $x : \kappa \vdash_{\mathtt{ST}} t : \mathtt{o}$ where t does not contain the non-deterministic choice, t is called *linear* (with respect to x) if x occurs exactly once in the call-by-name normal form of t. A pair of contexts $[\,] : \kappa \vdash_{\mathtt{ST}} C : \mathtt{o}$ and $[\,] : \kappa \vdash_{\mathtt{ST}} D : \kappa$ is called *linear* if $x : \kappa \vdash_{\mathtt{ST}} C[D^i[x]] : \mathtt{o}$ is linear for any $i \geq 0$ where x is a fresh variable that is not captured by the context applications.

▶ **Definition 3** (Higher-Order Grammar)**.** A *higher-order grammar* (or *grammar* for short) is a quadruple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$, where (i) Σ is a ranked alphabet; (ii) $\mathcal{N}$ is a map from a finite set of non-terminals to their types; (iii) $\mathcal{R}$ is a finite set of *rewriting rules* of the form $A \to \lambda x_1.\cdots\lambda x_\ell.t$, where $\mathcal{N}(A) = \kappa_1 \to \cdots \to \kappa_\ell \to \mathtt{o}$, t is an applicative term, and $\mathcal{N}, x_1 : \kappa_1, \ldots, x_\ell : \kappa_\ell \vdash_{\mathtt{ST}} t : \mathtt{o}$ holds; (iv) S is a non-terminal called the *start symbol*, and $\mathcal{N}(S) = \mathtt{o}$. The *order* of a grammar $\mathcal{G}$ is the largest order of the types of non-terminals. We sometimes write $\Sigma_\mathcal{G}, \mathcal{N}_\mathcal{G}, \mathcal{R}_\mathcal{G}, S_\mathcal{G}$ for the four components of $\mathcal{G}$. We often write $A\,x_1 \cdots x_k \to t$ for the rule $A \to \lambda x_1.\cdots\lambda x_k.t$.

For a grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, the rewriting relation $\longrightarrow_\mathcal{G}$ is defined by:

$$\frac{(A \to \lambda x_1.\cdots\lambda x_k.t) \in \mathcal{R}}{A\,t_1 \cdots t_k \longrightarrow_\mathcal{G} [t_1/x_1, \ldots, t_k/x_k]t} \qquad \frac{t_i \longrightarrow_\mathcal{G} t_i' \qquad i \in \{1, \ldots, k\} \qquad \Sigma(a) = k}{a\,t_1 \cdots t_k \longrightarrow_\mathcal{G} a\,t_1 \cdots t_{i-1}\,t_i'\,t_{i+1} \cdots t_k}$$

We write $\longrightarrow_\mathcal{G}^*$ for the reflexive transitive closure of $\longrightarrow_\mathcal{G}$. The *tree language generated by* $\mathcal{G}$, written $\mathcal{L}(\mathcal{G})$, is the set $\{\pi \mid S \longrightarrow_\mathcal{G}^* \pi\}$.

▶ Remark. An order-n grammar can also be represented as a ground closed order-n $\lambda^{\to,+}$-term extended with the Y-combinator such that $Y_\kappa x.t \longrightarrow [Y_\kappa x.t/x]t$. Conversely, any ground closed order-n $\lambda^{\to,+}$-term (extended with Y) can be represented as an equivalent order-n grammar.

The grammars defined above may also be viewed as generators of word languages.

▶ **Definition 4** (Word Alphabet / br-Alphabet)**.** We call a ranked alphabet Σ a *word alphabet* if it has a special nullary terminal $\mathtt{e}$ and all the other terminals have arity 1; also we call a grammar $\mathcal{G}$ a *word grammar* if its alphabet is a word alphabet. For a tree $\pi = a_1(\cdots(a_n\,\mathtt{e})\cdots)$ of a word grammar, we define $\mathbf{word}(\pi) = a_1 \cdots a_n$. The *word language* generated by a word grammar $\mathcal{G}$, written $\mathcal{L}_\mathtt{w}(\mathcal{G})$, is $\{\mathbf{word}(\pi) \mid \pi \in \mathcal{L}(\mathcal{G})\}$.

The frontier word of a tree π, written $\mathbf{leaves}(\pi)$, is the sequence of symbols in the leaves of π. It is defined inductively by: $\mathbf{leaves}(a) = a$ when $\Sigma(a) = 0$, and $\mathbf{leaves}(a\,\pi_1 \cdots \pi_k) = \mathbf{leaves}(\pi_1) \cdots \mathbf{leaves}(\pi_k)$ when $\Sigma(a) = k > 0$. The *frontier language* generated by $\mathcal{G}$, written

$\mathcal{L}_{\mathtt{leaf}}(\mathcal{G})$, is the set: $\{\mathbf{leaves}(\pi) \mid S \longrightarrow^*_{\mathcal{G}} \pi\}$. A $\mathtt{br}$-*alphabet* is a ranked alphabet such that it has a special binary constant $\mathtt{br}$ and a special nullary constant $\mathtt{e}$ and the other constants are nullary. We consider $\mathtt{e}$ as the empty word ε: for a grammar with a $\mathtt{br}$-alphabet, we also define $\mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}) := (\mathcal{L}_{\mathtt{leaf}}(\mathcal{G}) \setminus \{\mathtt{e}\}) \cup \{\varepsilon \mid \mathtt{e} \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\}$. We call a tree π an $\mathtt{e}$-*free* $\mathtt{br}$-*tree* if it is a tree of some $\mathtt{br}$-alphabet but does not contain $\mathtt{e}$.

We note that the classes of order-0, order-1, and order-2 word languages coincide with those of regular, context-free, and indexed languages, respectively [26].

2.2 Homeomorphic Embedding and Kruskal's Tree Theorem

In our main theorem, we use the notion of homeomorphic embedding for trees.

▶ **Definition 5** (Homeomorphic Embedding). Let Σ be an arbitrary ranked alphabet. The *homeomorphic embedding* order $\preceq$ between Σ-ranked trees[5] is inductively defined by the following rules:

$$\frac{\pi_i \preceq \pi'_i \quad (\text{for all } i \le k)}{a\,\pi_1 \cdots \pi_k \preceq a\,\pi'_1 \cdots \pi'_k}(k = \Sigma(a)) \qquad \frac{\pi \preceq \pi_i}{\pi \preceq a\,\pi_1 \cdots \pi_k}(k = \Sigma(a) > 0,\ i \in \{1, \ldots, k\})$$

For example, $\mathtt{br}\,\mathtt{a}\,\mathtt{b} \preceq \mathtt{br}\,(\mathtt{br}\,\mathtt{a}\,\mathtt{c})\,\mathtt{b}$. We extend $\preceq$ to words: for $w = a_1 \cdots a_n$ and $w' = a'_1 \cdots a'_{n'}$, we define $w \preceq w'$ if $a_1(\cdots(a_n(\mathtt{e}))) \preceq a'_1(\cdots(a'_{n'}(\mathtt{e})))$, where a_i and a'_i are regarded as unary constants and $\mathtt{e}$ is a nullary constant (this order on words is nothing but the (scattered) subsequence relation). We write $\pi \prec \pi'$ if $\pi \preceq \pi'$ and $\pi' \not\preceq \pi$.

Next we explain a basic property on $\preceq$, Kruskal's tree theorem. A *quasi-order* (also called a *pre-order*) is a reflexive and transitive relation. A *well quasi-order* on a set S is a quasi-order $\le$ on S such that for any infinite sequence $(s_i)_i$ of elements in S there exist $j < k$ such that $s_j \le s_k$.

▶ **Proposition 6** (Kruskal's Tree Theorem [17]). *For any (finite) ranked alphabet Σ, the homeomorphic embedding $\preceq$ on Σ-ranked trees is a well quasi-order.*

2.3 Conjecture and Pumping Lemma for Higher-order Grammars

As explained in Section 1, our pumping lemma makes use of a conjecture on "higher-order" Kruskal's tree theorem, which is stated below.

▶ **Conjecture 7.** *There exists a family $(\preceq_\kappa)_\kappa$ of relations indexed by simple types such that*

- *$\preceq_\kappa$ is a well quasi-order on the set of closed $\lambda^{\to}$-terms of type κ modulo $\beta\eta$-equivalence; i.e., for an infinite sequence $t_1, t_2, \ldots$ of closed $\lambda^{\to}$-terms of type κ, there exist $i < j$ such that $t_i \preceq_\kappa t_j$.*
- *$\preceq_{\mathtt{o}}$ is a conservative extension of $\preceq$, i.e., $t \preceq_{\mathtt{o}} t'$ if and only if $\mathcal{T}(t) \preceq \mathcal{T}(t')$.*
- *$(\preceq_\kappa)_\kappa$ is closed under applications, i.e., if $t \preceq_{\kappa_1 \to \kappa_2} t'$ and $s \preceq_{\kappa_1} s'$ then $t\,s \preceq_{\kappa_2} t'\,s'$.*

A candidate of $(\preceq_\kappa)_\kappa$ would be the logical relation induced from $\preceq$. Indeed, if we choose the logical relation as $(\preceq_\kappa)_\kappa$, the above conjecture holds up to order-2 (see Theorem 18 in Section 6).

Actually, for our pumping lemma, the following, slightly weaker property called the *periodicity* is sufficient.

[5] In the usual definition, a quasi order on labels (tree constructors) is assumed. Here we fix the quasi-order on labels to the identity relation.

▶ **Conjecture 8** (Periodicity). *There exists a family $(\preceq_\kappa)_\kappa$ indexed by simple types such that*

- *$\preceq_\kappa$ is a quasi-order on the set of closed $\lambda^{\to}$-terms of type κ modulo $\beta\eta$-equivalence.*
- *for any $\vdash_{\mathtt{ST}} t : \kappa \to \kappa$ and $\vdash_{\mathtt{ST}} s : \kappa$, there exist $i, j > 0$ such that*

$$t^i\ s \preceq_\kappa t^{i+j}\ s \preceq_\kappa t^{i+2j}\ s \preceq_\kappa \cdots .$$

- *$\preceq_{\mathtt{o}}$ is a conservative extension of $\preceq$.*
- *$(\preceq_\kappa)_\kappa$ is closed under applications.*

Note that Conjecture 7 implies Conjecture 8, since if the former holds, for the infinite sequence $(t^i\ s)_i$, there exist $i < i + j$ such that $t^i\ s \preceq_\kappa t^{i+j}\ s$, and then by the monotonicity of $u \mapsto t^j\ u$, we have $t^{i+kj}\ s \preceq_\kappa t^{i+(k+1)j}\ s$ for any $k \geq 0$.

We can now state our pumping lemma.

▶ **Theorem 9** (Pumping Lemma). *Assume that Conjecture 8 holds. Then, for any order-n tree grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G})$ is infinite, there exist an infinite sequence of trees $\pi_0, \pi_1, \pi_2, \ldots \in \mathcal{L}(\mathcal{G})$, and constants c, d such that:* (i) $\pi_0 \prec \pi_1 \prec \pi_2 \prec \cdots$, *and* (ii) $|\pi_i| \leq \mathbf{exp}_n(ci + d)$ *for each $i \geq 0$. Furthermore, we can drop the assumption on Conjecture 8 when $\mathcal{G}$ is of order up to 2.*

By the correspondence between order-n tree grammars and order-$(n+1)$ grammars [4, 1], we also have:

▶ **Corollary 10** (Pumping Lemma for Word Languages). *Assume that Conjecture 8 holds. Then, for any order-n word grammar $\mathcal{G}$ (where $n \geq 1$) such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$ is infinite, there exist an infinite sequence of words $w_0, w_1, w_2, \ldots \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$, and constants c, d such that:* (i) $w_0 \prec w_1 \prec w_2 \prec \cdots$, *and* (ii) $|w_i| \leq \mathbf{exp}_{n-1}(ci + d)$ *for each $i \geq 0$. Furthermore, we can drop the assumption on Conjecture 8 when $\mathcal{G}$ is of order up to 3.*

We sketch the overall structure of the proof of Theorem 9 below. Let $\mathcal{G}$ be an order-n tree grammar. By using the recent type system of Parys [23], if $\mathcal{L}(\mathcal{G})$ is infinite, we can construct order-n linear $\lambda^{\to}$-contexts C, D and an order-n $\lambda^{\to}$-term t such that $\{\mathcal{T}(C[D^i[t]]) \mid i \geq 0\}$ $(\subseteq \mathcal{L}(\mathcal{G}))$ is infinite. It then suffices to show that there exist constants p and q such that $\mathcal{T}(C[D^p[t]]) \prec \mathcal{T}(C[D^{p+q}[t]]) \prec \mathcal{T}(C[D^{p+2q}[t]]) \prec \cdots$. The bound $\mathcal{T}(C[D^{p+iq}]) \leq \mathbf{exp}_n(c + id)$ would then follow immediately from the standard result on an upper-bound on the size of β-normal forms. Actually, assuming Conjecture 8, we can easily deduce $\mathcal{T}(C[D^p[t]]) \preceq \mathcal{T}(C[D^{p+q}[t]]) \preceq \mathcal{T}(C[D^{p+2q}[t]]) \preceq \cdots$. Thus, the main remaining difficulty is to show that the "strict" inequality holds periodically. To this end, we prove it by induction on the order, by making use of three ingredients: an extension of the result of Parys' type system (again) [23], an extension of our previous work on a translation from word languages to tree languages [1], and Conjecture 8. In Sections 3 and 4, we derive corollaries from the results of Parys' and our previous work respectively. We then provide the proof of Theorem 9 (except the statement "Furthermore, ...") in Section 5. We then, in Section 6, discharge the assumption on Conjecture 8 for order up to 2, by proving Conjecture 7 for order up to 2.

3 Corollaries of Parys' Results

Parys [23] developed an intersection type system with judgments of the form $\Gamma \vdash s : \tau \triangleright c$, where s is a term of a simply-typed, infinitary λ-calculus (that corresponds to the λY-calculus) extended with choice, and c is a natural number. He proved that for any order-n closed ground term s, (i) $\emptyset \vdash s : \tau \triangleright c$ implies that s can be reduced to a tree π such that $c \leq |\pi|$, and (ii) if s can be reduced to a tree π, then $\emptyset \vdash s : \tau \triangleright c$ holds for some c such that $|\pi| \leq \mathbf{exp}_n(c)$.

Let $\mathcal{G}$ be an order-n tree grammar and S be its start symbol. By Parys' result,[6] if $\mathcal{L}(\mathcal{G})$ is infinite, there exists a derivation for $\emptyset \vdash S : \mathtt{o} \triangleright c_1 + c_2 + c_3$ in which $\Theta \vdash A : \gamma \triangleright c_1 + c_2$ is derived from $\Theta \vdash A : \gamma \triangleright c_1$ for some non-terminal A. Thus, by "pumping" the derivation of $\Theta \vdash A : \gamma \triangleright c_1 + c_2$ from $\Theta \vdash A : \gamma \triangleright c_1$, we obtain a derivation for $\emptyset \vdash S : \mathtt{o} \triangleright c_1 + kc_2 + c_3$ for any $k \geq 0$. From the derivation, we obtain a $\lambda^{\to}$-term t and $\lambda^{\to}$-contexts C, D of at most order-n, such that $C[D^k[t]]$ generates a tree π_k such that $c_1 + kc_2 + c_3 \leq |\pi_k|$. By further refining the argument above (see the full version for details), we can also ensure that the pair (C, D) is linear. Thus, we obtain the following lemma.

▶ **Lemma 11.** *Given an order-n tree grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G})$ is infinite, there exist order-n linear $\lambda^{\to}$-contexts C, D, and an order-n $\lambda^{\to}$-term t such that:*
1. $\{\mathcal{T}(C[D^k[t]]) \mid k \geq 1\} \subseteq \mathcal{L}(\mathcal{G})$,
2. $\{\mathcal{T}(C[D^{\ell_k}[t]]) \mid k \geq 1\}$ *is infinite for any strictly increasing sequence* $(\ell_k)_k$.

By slightly modifying Parys' type system, we can also reason about the length of a particular path of a tree. Let us annotate each constructor a as $a^{\langle i \rangle}$, where $0 \leq i \leq \Sigma(a)$. We call i a *direction*. We define $|\pi|_{\mathrm{p}}$ by:

$$|a^{\langle 0 \rangle}\, \pi_1 \cdots \pi_k|_{\mathrm{p}} = 1 \qquad |a^{\langle i \rangle}\, \pi_1 \cdots \pi_k|_{\mathrm{p}} = |\pi_i|_{\mathrm{p}} + 1 \quad (1 \leq i \leq k).$$

We define **rmdir** as the function that removes all the direction annotations.

▶ **Lemma 12.** *For any order-n linear $\lambda^{\to}$-contexts C, D and any order-n $\lambda^{\to}$-term t such that $\{\mathcal{T}(C[D^k[t]]) \mid k \geq 1\}$ is infinite, there exist direction-annotated order-n linear $\lambda^{\to}$-contexts G, H, a direction-annotated order-n $\lambda^{\to}$-term u, and $p, q > 0$ such that*
1. $\mathbf{rmdir}(\mathcal{T}(G[H^k[u]])) = \mathcal{T}(C[D^{pk+q}[t]])$ *for any $k \geq 1$,*
2. $\{|\mathcal{T}(G[H^{\ell_k}[u]])|_{\mathrm{p}} \mid k \geq 1\}$ *is infinite for any strictly increasing sequence* $(\ell_k)_k$.

4 Word to Frontier Transformation

We have an "order-decreasing" transformation [1] that transforms an order-$(n+1)$ word grammar $\mathcal{G}$ to an order-n tree grammar $\mathcal{G}'$ (with a $\mathtt{br}$-alphabet) such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$. We use this as a method for induction on order; this method was originally suggested by Damm [4] for safe languages.

The transformation in the present paper has been modified from the original one in [1]. On the one hand, the current transformation is a specialized version in that we apply the transformation only to $\lambda^{\to}$-terms instead of terms of (non-deterministic) grammars. On the other hand, the current transformation has been strengthened in that the transformation preserves linearity. Due to the preservation of linearity, a *single*-hole context is transformed to a *single*-hole context, and the uniqueness of an occurrence of $[\,]$ will be utilized for the calculation of the size of "pumped trees" in Lemma 16.

The definition of the current transformation is given just by translating the transformation rules in [1] by following the idea of the embedding of $\lambda^{\to}$-terms into grammars. For the detailed definition, see the full version. By using this transformation, we have:

▶ **Lemma 13.** *Given order-n $\lambda^{\to}$-contexts C, D, and an order-n $\lambda^{\to}$-term t such that the constants in C, D, t are in a word alphabet,*

[6] See Section 6 of [23]. Parys considered a λ-calculus with infinite regular terms, but the result can be easily adapted to terms of grammars.

- $\{\mathcal{T}(C[D^{\ell_i}[t]]) \mid i \geq 0\}$ *is infinite for any strictly increasing sequence* $(\ell_i)_i$*, and*
- C *and* D *are linear,*

there exist order-$(n-1)$ $\lambda^{\to}$*-contexts* G*,* H*, order-*$(n-1)$ $\lambda^{\to}$*-term* u*, and some constant numbers* $c, d \geq 1$ *such that*

- *the constants in* G*,* H*,* u *are in a* `br`*-alphabet*
- *for* $i \geq 0$*,* $\mathcal{T}(G[H^i[u]])$ *is either an* `e`*-free* `br`*-tree or* `e`*, and*

$$\mathbf{word}(\mathcal{T}(C[D^{ci+d}[t]])) = \begin{cases} \varepsilon & (\mathcal{T}(G[H^i[u]]) = \mathtt{e}) \\ \boldsymbol{leaves}(\mathcal{T}(G[H^i[u]])) & (\mathcal{T}(G[H^i[u]]) \neq \mathtt{e}) \end{cases}$$

- G *and* H *are linear.*

Proof. The preservation of meaning (the second condition) follows as a corollary of a theorem in [1]. Also, the preservation of linearity (the third condition) can be proved in a manner similar to the proof of the preservation of meaning in [1], using a kind of subject-reduction. See the full version for the detail. ◀

5 Proof of the Main Theorem

We first prepare some lemmas.

▶ **Lemma 14.** *For* `e`*-free* `br`*-trees* π *and* π'*, if* $\pi \prec \pi'$ *then* $\boldsymbol{leaves}(\pi) \prec \boldsymbol{leaves}(\pi')$*.*

Proof. We can show that $\pi \preceq \pi'$ implies $\mathbf{leaves}(\pi) \preceq \mathbf{leaves}(\pi')$ and then the statement, both by straightforward induction on the derivation of $\pi \preceq \pi'$. ◀

▶ Remark. The above lemma does not necessarily hold for an arbitrary ranked alphabet, especially that with a unary constant; e.g., $\mathtt{a}\,\mathtt{e} \prec \mathtt{a}\,(\mathtt{a}\,\mathtt{e})$ but their leaves are both `e`. Also, it does not hold if a tree contains `e` and if we regard `e` as ε in the leaves word; e.g., for $\mathtt{br}\,\mathtt{a}\,\mathtt{b} \prec \mathtt{br}\,(\mathtt{br}\,\mathtt{a}\,\mathtt{e})\,\mathtt{b}$, their leaves are $\mathtt{ab} \prec \mathtt{aeb}$, but if we regard `e` as ε then $\mathtt{ab} \not\prec \mathtt{ab}$.

▶ **Lemma 15.** *For direction-annotated trees* π *and* π'*, if* $\pi \prec \pi'$ *then* $\mathbf{rmdir}(\pi) \prec \mathbf{rmdir}(\pi')$*.*

Proof. We can show that $\pi \preceq \pi'$ implies $\mathbf{rmdir}(\pi) \preceq \mathbf{rmdir}(\pi')$ and then the statement, both by straightforward induction on the derivation of $\pi \preceq \pi'$. ◀

Now, we prove the following lemma (Lemma 16) by the induction on order. Theorem 9 (except the last statement) will then follow as an immediate corollary of Lemmas 11 and 16.

▶ **Lemma 16.** *Assume that the statement of Conjecture 8 is true. For any order-*n *linear* $\lambda^{\to}$*-contexts* C, D *and any order-*n $\lambda^{\to}$*-term* t *such that* $\{\mathcal{T}(C[D^i[t]]) \mid i \geq 1\}$ *is infinite, there exist* c*,* d*,* j*,* $k \geq 1$ *such that*

- $\mathcal{T}(C[D^j[t]]) \prec \mathcal{T}(C[D^{j+k}[t]]) \prec \mathcal{T}(C[D^{j+2k}[t]]) \prec \cdots$
- $|\mathcal{T}(C[D^{j+ik}[t]])| \leq \mathbf{exp}_n(ci+d) \qquad (i = 0, 1, \dots)$

Proof. The proof proceeds by induction on n. The case $n = 0$ is clear, and we discuss the case $n > 0$ below. By Lemma 12, from C, D, and t, we obtain direction-annotated order-n linear $\lambda^{\to}$-contexts G, H, a direction-annotated order-n $\lambda^{\to}$-term u, and j_0, $k_0 > 0$ such that

$$\mathbf{rmdir}(\mathcal{T}(G[H^i[u]])) = \mathcal{T}(C[D^{j_0+ik_0}[t]]) \text{ for any } i \geq 1 \tag{1}$$

$$\{|\mathcal{T}(G[H^{\ell_i}[u]])|_{\mathrm{p}} \mid i \geq 1\} \text{ is infinite for any strictly increasing sequence } (\ell_i)_i. \tag{2}$$

Next we transform G, H, and u by choosing a path according to directions, i.e., we define G_{p}, H_{p}, and u_{p} as the contexts/term obtained from G, H, and u by replacing each $a^{\langle i\rangle}$ with: (i) $\lambda x_1 \dots x_\ell.a_i x_i$ if $i > 0$ or (ii) $\lambda x_1 \dots x_\ell.\mathtt{e}$ if $i = 0$, where $\ell = \Sigma(a)$ and a_i is a fresh unary constant. For any $i \geq 0$,

$$|\mathcal{T}(G[H^i[u]])|_{\mathrm{p}} = |\mathbf{word}(\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[u_{\mathrm{p}}]]))| + 1. \tag{3}$$

We also define a function **path** on trees annotated with directions, by the following induction: $\mathbf{path}(a^{\langle i\rangle}\,\pi_1 \cdots \pi_\ell) = a_i\,\mathbf{path}(\pi_i)$ if $i > 0$ and $\mathbf{path}(a^{\langle 0\rangle}\,\pi_1 \cdots \pi_\ell) = \mathtt{e}$. Then for any $i \geq 0$,

$$\mathbf{path}(\mathcal{T}(G[H^i[u]])) = \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[u_{\mathrm{p}}]]). \tag{4}$$

By (2) and (3), $\{\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{\ell_i}[u_{\mathrm{p}}]]) \mid i \geq 0\}$ is infinite for any strictly increasing sequence $(\ell_i)_i$. Also, the transformation from G, H to G_{p}, H_{p} preserves the linearity, because: let N be the normal form of $G[H^i[x]]$ where x is fresh, and N_{p} be the term obtained by applying this transformation to N; then $G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[x]] \longrightarrow^* N_{\mathrm{p}}$, and by the infiniteness of $\{\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[u_{\mathrm{p}}]]) \mid i \geq 0\}$, N_{p} must contain x, which implies N_{p} is a linear normal form.

Now we decrease the order by using the transformation in Section 4. By Lemma 13 to G_{p}, H_{p}, and u_{p}, there exist order-$(n-1)$ linear $\lambda^{\to}$-contexts G_l, H_l, an order-$(n-1)$ $\lambda^{\to}$-term u_l, and some constant numbers c', $d' \geq 1$ such that, for any $i \geq 0$, $\mathcal{T}(G_l[{H_l}^i[u_l]])$ is either an $\mathtt{e}$-free $\mathtt{br}$-tree or $\mathtt{e}$, and

$$\mathbf{word}(\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'i+d'}[u_{\mathrm{p}}]])) = \begin{cases} \varepsilon & (\mathcal{T}(G_l[{H_l}^i[u_l]]) = \mathtt{e}) \\ \mathbf{leaves}(\mathcal{T}(G_l[{H_l}^i[u_l]])) & (\mathcal{T}(G_l[{H_l}^i[u_l]]) \neq \mathtt{e}). \end{cases} \tag{5}$$

By (2), (3), and (5), $\{\mathcal{T}(G_l[{H_l}^i[u_l]]) \mid i \geq 1\}$ is also infinite.

By the induction hypothesis, there exist j_1 and k_1 such that

$$\mathcal{T}(G_l[{H_l}^{j_1}[u_l]]) \prec \mathcal{T}(G_l[{H_l}^{j_1+k_1}[u_l]]) \prec \mathcal{T}(G_l[{H_l}^{j_1+2k_1}[u_l]]) \prec \cdots.$$

Hence by Lemma 14, we have

$$\mathbf{leaves}(\mathcal{T}(G_l[{H_l}^{j_1}[u_l]])) \prec \mathbf{leaves}(\mathcal{T}(G_l[{H_l}^{j_1+k_1}[u_l]])) \prec \mathbf{leaves}(\mathcal{T}(G_l[{H_l}^{j_1+2k_1}[u_l]])) \prec \cdots.$$

Then by (5), we have

$$\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'j_1+d'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'(j_1+k_1)+d'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'(j_1+2k_1)+d'}[u_{\mathrm{p}}]]) \prec \cdots.$$

Let $j_1' = c'j_1 + d'$ and $k_1' = c'k_1$; then

$$\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_1'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_1'+k_1'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_1'+2k_1'}[u_{\mathrm{p}}]]) \prec \cdots. \tag{6}$$

Now, by Conjecture 8, there exist $j_2 \geq 0$ and $k_2 > 0$ such that

$$H^{j_2}[u] \preceq_\kappa H^{j_2+k_2}[u] \preceq_\kappa H^{j_2+2k_2}[u] \preceq_\kappa \cdots. \tag{7}$$

Let j_3 be the least j_3 such that $j_3 = j_1' + i_3\,k_1' = j_2 + m_0$ for some i_3 and m_0, and k_3 be the least common multiple of k_1' and k_2, whence $k_3 = m_1 k_1' = m_2 k_2$ for some m_1 and m_2. Then since the mapping $s \mapsto \mathcal{T}(G[H^{m_0}[s]])$ is monotonic, from (7) we have:

$$\mathcal{T}(G[H^{j_3}[u]]) \preceq \mathcal{T}(G[H^{j_3+k_2}[u]]) \preceq \mathcal{T}(G[H^{j_3+2k_2}[u]]) \preceq \cdots.$$

Since $j_3 + ik_3 = j_3 + (im_2)k_2$, we have

$$\mathcal{T}(G[H^{j_3}[u]]) \preceq \mathcal{T}(G[H^{j_3+k_3}[u]]) \preceq \mathcal{T}(G[H^{j_3+2k_3}[u]]) \preceq \cdots. \tag{8}$$

Also, since $j_3 + ik_3 = j'_1 + (i_3 + im_1)k'_1$, from (6) we have

$$\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_3}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_3+k_3}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_3+2k_3}[u_{\mathrm{p}}]]) \prec \cdots . \tag{9}$$

Thus, from (4), (8), and (9) we obtain

$$\mathcal{T}(G[H^{j_3}[u]]) \prec \mathcal{T}(G[H^{j_3+k_3}[u]]) \prec \mathcal{T}(G[H^{j_3+2k_3}[u]]) \prec \cdots . \tag{10}$$

By applying **rmdir** to this sequence, and by (1) and Lemma 15, we have

$$\mathcal{T}(C[D^{j_0+j_3k_0}[t]]) \prec \mathcal{T}(C[D^{j_0+(j_3+k_3)k_0}[t]]) \prec \mathcal{T}(C[D^{j_0+(j_3+2k_3)k_0}[t]]) \prec \cdots . \tag{11}$$

We define $j = j_0 + k_0 j_3$ and $k = k_0 k_3$; then we obtain

$$\mathcal{T}(C[D^{j}[t]]) \prec \mathcal{T}(C[D^{j+k}[t]]) \prec \mathcal{T}(C[D^{j+2k}[t]]) \prec \cdots .$$

Finally, we show that $|\mathcal{T}(C[D^{j+ik}[t]])| \leq \mathbf{exp}_n(ci + d)$ for some c and d. Since C and D are single-hole contexts, $|C[D^{j+ik}[t]]| = |C| + (j + ik)|D| + |t|$. Let $c = k|D|$ and $d = |C| + j|D| + |t|$; then $|C[D^{j+ik}[t]]| = ci + d$. It is well-known that, for an order-n $\lambda^{\to}$-term s, we have $|\mathcal{T}(s)| \leq \mathbf{exp}_n(|s|)$ (see, e.g., [25, Lemma 3]). Thus, we have $|\mathcal{T}(C[D^{j+ik}[t]])| \leq \mathbf{exp}_n(ci + d)$. ◀

The step obtaining (10) (the steps using Lemma 14 and obtaining (11), resp.) indicates why we need to require $\mathcal{T}(C[D^{j+ik}[t]]) \prec \mathcal{T}(C[D^{j+i'k}[t]])$ for any $i < i'$ rather than $|\mathcal{T}(C[D^{j+ik}[t]])| < |\mathcal{T}(C[D^{j+i'k}[t]])|$ ($\mathcal{T}(C[D^{j+ik}[t]]) \neq \mathcal{T}(C[D^{j+i'k}[t]])$, resp.) to make the induction work.

6 Second-order Kruskal's theorem

In this section, we prove Conjecture 7 (hence also Conjecture 8) up to order-2. First, we extend the homeomorphic embedding $\preceq$ on trees to a family of relations $\preceq_\kappa$ by using logical relation: (i) $t_1 \preceq_{\mathsf{o}} t_2$ if $\emptyset \vdash_{\mathtt{ST}} t_1 : \mathsf{o}$, $\emptyset \vdash_{\mathtt{ST}} t_2 : \mathsf{o}$, and $\mathcal{T}(t_1) \preceq \mathcal{T}(t_2)$. (ii) $t_1 \preceq_{\kappa_1 \to \kappa_2} t_2$ if $\emptyset \vdash_{\mathtt{ST}} t_1 : \kappa_1 \to \kappa_2$, $\emptyset \vdash_{\mathtt{ST}} t_2 : \kappa_1 \to \kappa_2$, and $t_1 s_1 \preceq_{\kappa_2} t_2 s_2$ holds for every s_1, s_2 such that $s_1 \preceq_{\kappa_1} s_2$. We often omit the subscript κ and just write $\preceq$ for $\preceq_\kappa$. We also write $x_1 : \kappa_1, \ldots, x_k : \kappa_k \models t \preceq_\kappa t'$ if $[s_1/x_1, \ldots, s_k/x_k]t \preceq_\kappa [s'_1/x_1, \ldots, s'_k/x_k]t'$ for every $s_1, \ldots, s_k, s'_1, \ldots, s'_k$ such that $s_i \preceq_{\kappa_i} s'_i$.

The relation $\preceq_\kappa$ is well-defined for $\beta\eta$-equivalence classes, and by the abstraction lemma of logical relation, it turns out that the relation $\preceq_\kappa$ is a pre-order for any κ (see the full version for these). Note that the relation is also preserved by applications by the definition of the logical relation. It remains to show that $\preceq_\kappa$ is a well quasi-order for κ of order up to 2.

For ℓ-ary terminal a and $k \geq \ell$, we write $\mathbf{CTerms}_{a,k}$ for the set of terms

$$\{\lambda x_1.\cdots\lambda x_k.a\, x_{i_1}\, \ldots\, x_{i_\ell} \mid i_1 \cdots i_\ell \text{ is a subsequence of } 1 \cdots k\}.$$

We define $\mathsf{o}^0 \to \mathsf{o} := \mathsf{o}$ and $\mathsf{o}^{n+1} \to \mathsf{o} := \mathsf{o} \to (\mathsf{o}^n \to \mathsf{o})$.

The following lemma allows us to reduce $t \preceq_\kappa t'$ on any order-2 type κ to (finitely many instances of) that on order-0 type o.

▶ **Lemma 17.** *Let Σ be a ranked alphabet; κ be $(\mathsf{o}^{k_1} \to \mathsf{o}) \to \cdots \to (\mathsf{o}^{k_m} \to \mathsf{o}) \to \mathsf{o}$; a_i^j be a j-ary terminal not in Σ for $1 \leq i \leq m$ and $0 \leq j \leq k_i$; and t, t' be $\lambda^{\to}$-terms whose type is κ and whose terminals are in Σ. Then $t \preceq_\kappa t'$ if and only if $t\, u_1\, \ldots\, u_m \preceq_{\mathsf{o}} t'\, u_1\, \ldots\, u_m$ for every $u_i \in \cup_{j \leq k_i} \mathbf{CTerms}_{a_i^j, k_i}$.*

Proof. The "only if" direction is trivial by the definition of $\preceq_\kappa$. To show the opposite, assume the latter holds. We need to show that $t\,s_1\,\ldots\,s_m \preceq_{\mathsf{o}} t'\,s_1\,\ldots\,s_m$ holds for every combination of $s_1,\ldots,s_m$ such that $\vdash_{\mathtt{ST}} s_i : \kappa_i$ for each i. Without loss of generality, we can assume that t, t', $s_1,\ldots,s_m$ are $\beta\eta$ long normal forms, and hence that

$$\begin{array}{lll} t = \lambda f_1.\cdots\lambda f_m.t_0 & f_1:\mathsf{o}^{k_1}\to\mathsf{o},\ldots,f_m:\mathsf{o}^{k_m}\to\mathsf{o}\vdash_{\mathtt{ST}} t_0:\mathsf{o} & \\ t' = \lambda f_1.\cdots\lambda f_m.t'_0 & f_1:\mathsf{o}^{k_1}\to\mathsf{o},\ldots,f_m:\mathsf{o}^{k_m}\to\mathsf{o}\vdash_{\mathtt{ST}} t'_0:\mathsf{o} & \\ s_i = \lambda x_1.\ldots\lambda x_{k_i}.s_{i,0} & x_1:\mathsf{o},\ldots,x_{k_i}:\mathsf{o}\vdash_{\mathtt{ST}} s_{i,0}:\mathsf{o} & \text{(for each } i\text{)} \end{array}$$

For each $i \le m$, let $\mathbf{FV}(s_{i,0}) = \{x_{q(i,1)},\ldots,x_{q(i,\ell_i)}\}$, and $u_i \in \mathbf{CTerms}_{a_i^{\ell_i},k_i}$ be the term $\lambda x_1.\cdots\lambda x_{k_i}.a_i^{\ell_i}\,x_{q(i,1)}\,\cdots\,x_{q(i,\ell_i)}$. Let θ and θ' be the substitutions $[u_1/f_1,\ldots,u_m/f_m]$ and $[s_1/f_1,\ldots,s_m/f_m]$ respectively. It suffices to show that $\theta t_0 \preceq_{\mathsf{o}} \theta t'_0$ implies $\theta' t_0 \preceq_{\mathsf{o}} \theta' t'_0$, which we prove by induction on $|t'_0|$.

By the condition $f_1:\mathsf{o}^{k_1}\to\mathsf{o},\ldots,f_m:\mathsf{o}^{k_m}\to\mathsf{o}\vdash_{\mathtt{ST}} t_0:\mathsf{o}$, t_0 must be of the form $h\,t_1\,\cdots\,t_\ell$ where h is f_i or a terminal a in Σ, and ℓ may be 0. Then we have

$$\mathcal{T}(\theta t_0) = \begin{cases} a\mathcal{T}(\theta t_1)\cdots\mathcal{T}(\theta t_\ell) & (h=a) \\ a_i^{\ell_i}\mathcal{T}(\theta t_{q(i,1)})\cdots\mathcal{T}(\theta t_{q(i,\ell_i)}) & (h=f_i) \end{cases}$$

Similarly, t'_0 must be of the form $h'\,t'_1\,\cdots\,t'_{\ell'}$ and the corresponding equality on $\mathcal{T}(\theta t'_0)$ holds. By the assumption $\theta t_0 \preceq_{\mathsf{o}} \theta t'_0$, we have $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_0)$. We perform case analysis on the rule used for deriving $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_0)$ (recall Definition 5).

- Case of the first rule: In this case, the roots of $\mathcal{T}(\theta t_0)$ and $\mathcal{T}(\theta t'_0)$ are the same and hence $h = h'$ and $\ell = \ell'$. We further perform case analysis on h.
  - Case $h = a$: For $1 \le j \le \ell$, since $\mathcal{T}(\theta t_j) \preceq \mathcal{T}(\theta t'_j)$, by induction hypothesis, we have $\theta' t_j \preceq_{\mathsf{o}} \theta' t'_j$. Hence $\theta' t_0 \preceq_{\mathsf{o}} \theta' t'_0$.
  - Case $h = f_i$: For $1 \le j \le \ell_i$, since $\mathcal{T}(\theta t_{q(i,j)}) \preceq \mathcal{T}(\theta t'_{q(i,j)})$, by induction hypothesis, we have $\theta' t_{q(i,j)} \preceq_{\mathsf{o}} \theta' t'_{q(i,j)}$. Hence, $[\theta' t_{q(i,j)}/x_{q(i,j)}]_{j\le\ell_i} s_{i,0} \preceq_{\mathsf{o}} [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j\le\ell_i} s_{i,0}$. By the definition of $q(i,j)$, $\theta' t_0 \longrightarrow [\theta' t_j/x_j]_{j\le k_i} s_{i,0} = [\theta' t_{q(i,j)}/x_{q(i,j)}]_{j\le\ell_i} s_{i,0}$, and similarly, $\theta' t'_0 \longrightarrow [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j\le\ell_i} s_{i,0}$; hence we have $\theta' t_0 \preceq_{\mathsf{o}} \theta' t'_0$.
- Case of the second rule: We further perform case analysis on h'.
  - Case $h' = a$: We have $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_p)$ for some $1 \le p \le \ell'$. Hence by induction hypothesis, we have $\theta' t_0 \preceq_{\mathsf{o}} \theta' t'_p$, and then $\theta' t_0 \preceq_{\mathsf{o}} \theta' t'_0$.
  - Case $h' = f_i$: We have $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_{q(i,p)})$ for some $1 \le p \le \ell_i$. Hence by induction hypothesis, we have $\theta' t_0 \preceq_{\mathsf{o}} \theta' t'_{q(i,p)}$. Also, by the definition of $q(i,p)$, $x_{q(i,p)}$ occurs in $s_{i,0}$. Since $s_{i,0}$ is a $\beta\eta$ long normal form of order-0, the order-0 variable $x_{q(i,p)}$ occurs as a leaf of $s_{i,0}$; hence $\mathcal{T}(\theta' t'_{q(i,p)}) \preceq [\mathcal{T}(\theta' t'_{q(i,j)})/x_{q(i,j)}]_{j\le\ell_i} s_{i,0}$. Therefore $\theta' t_0 \preceq_{\mathsf{o}} [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j\le\ell_i} s_{i,0}$. Since $\theta' t'_0 \longrightarrow [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j\le\ell_i} s_{i,0}$, we have $\theta' t_0 \preceq_{\mathsf{o}} \theta' t'_0$.

◀

As a corollary, we obtain a second-order version of Kruskal's tree theorem.

▶ **Theorem 18.** *Let Σ be a ranked alphabet, κ be an at most order-2 type, and $t_0, t_1, t_2, \ldots$ be an infinite sequence of $\lambda^{\to}$-terms whose type is κ and whose terminals are in Σ. Then, there exist $i < j$ such that $t_i \preceq_\kappa t_j$.*

Proof. Since κ is at most order-2, it must be of the form $(\mathsf{o}^{k_1}\to\mathsf{o})\to\cdots\to(\mathsf{o}^{k_m}\to\mathsf{o})\to\mathsf{o}$. Let a_i^j be a j-ary terminal not in Σ for $1 \le i \le m$ and $0 \le j \le k_i$; $(\cup_{j\le k_1}\mathbf{CTerms}_{a_1^j,k_1})\times\cdots\times(\cup_{j\le k_m}\mathbf{CTerms}_{a_m^j,k_m})$ be $\{(u_{1,1},\ldots,u_{1,m}),\ldots,(u_{p,1},\ldots,u_{p,m})\}$; b be a p-ary terminal not in $\Sigma\cup\{a_i^j \mid 1\le i\le m,\ 0\le j\le k_i\}$; and s_i be the term $b\,(t_i\,u_{1,1}\,\cdots\,u_{1,m})\,\cdots\,(t_i\,u_{p,1}\,\cdots\,u_{p,m})$

for each $i \in \{0, 1, 2, \ldots\}$. Since the set of terminals in $s_0, s_1, s_2, \ldots$ is finite, by Kruskal's tree theorem, there exist i, j such that $s_i \preceq_{\mathsf{o}} s_j$ and $i < j$. Since b occurs just at the root of s_k for each k, $s_i \preceq_{\mathsf{o}} s_j$ implies $t_i\, u_{k,1} \cdots u_{k,m} \preceq_{\mathsf{o}} t_j\, u_{k,1} \cdots u_{k,m}$ for every $k \in \{1, \ldots, p\}$. Thus, by Lemma 17, we have $t_i \preceq_\kappa t_j$ as required. ◀

7 Related Work

As mentioned in Section 1, to our knowledge, pumping lemmas for higher-order word languages have been established only up to order-2 [7], whereas we have proved (unconditionally) a pumping lemma for order-2 tree languages and order-3 word languages. Hayashi's pumping lemma for indexed languages (i.e., order-2 word languages) is already quite complex, and it is unclear how to generalize it to arbitrary orders. In contrast, our proof of a pumping lemma works for arbitrary orders, although it relies on the conjecture on higher-order Kruskal's tree theorem. Parys [21] and Kobayashi [12] studied pumping lemmas for collapsible pushdown automata and higher-order recursion schemes respectively. Unfortunately, they are not applicable to word/tree *languages* generated by (non-deterministic) grammars.

As also mentioned in Section 1, the strictness of hierarchy of higher-order word languages has already been shown by using a complexity argument [5, 8]. We can use our pumping lemma (if the conjecture is discharged) to obtain a simple alternative proof of the strictness, using the language $\{a^{\mathbf{exp}_n(k)} \mid k \geq 0\}$ as a witness of the separation between the classes of order-$(n+1)$ word languages and order-n word languages. In fact, the pumping lemma would imply that there is no order-n grammar that generates $\{a^{\mathbf{exp}_n(k)} \mid k \geq 0\}$, whereas an order-$(n+1)$ grammar that generates the same language can be easily constructed.

We are not aware of studies of the higher-order version of Kruskal's tree theorem (Conjecture 7) or the periodicity of tree functions expressed by the simply-typed λ-calculus (Conjecture 8), which seem to be of independent interest. Zaionc [27, 28] characterized the class of (first-order) word/tree functions definable in the simply-typed λ-calculus. To obtain higher-order Kruskal's tree theorem, we may need some characterization of *higher-order* definable tree functions instead.

We have heavily used the results of Parys' work [23] and our own previous work [1], which both use intersection types for studying properties of higher-order languages. Other uses of intersection types in studying higher-order grammars/languages are found in [10, 15, 22, 12, 3, 14, 13].

8 Conclusion

We have proved a pumping lemma for higher-order languages of arbitrary orders, modulo the assumption that a higher-order version of Kruskal's tree theorem holds. We have also proved the assumption indeed holds for the second-order case, yielding a pumping lemma for order-2 tree languages and order-3 word languages. Proving (or disproving) the higher-order Kruskal's tree theorem is left for future work.

Acknowledgments. We would like to thank Pawel Parys for discussions on his type system, and anonymous referees for useful comments.

References

1 Kazuyuki Asada and Naoki Kobayashi. On Word and Frontier Languages of Unsafe Higher-Order Grammars. In *43rd International Colloquium on Automata, Languages, and Pro-*

gramming (ICALP 2016), volume 55 of *LIPIcs*, pages 111:1–111:13. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

2 Yehoshua Bar-Hillel, Micha A. Perles, and Eli Shamir. On formal properties of simple phrase structure grammars. *Z. Phonetik Sprachwiss. und Kommunikat.*, 14:143–172, 1961.

3 Lorenzo Clemente, Pawel Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recusion schemes is decidable. In *Proceedings of LICS 2016*, 2016.

4 Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982.

5 Joost Engelfriet. Iterated stack automata and complexity classes. *Info. Comput.*, 95(1):21–75, 1991.

6 Joost Engelfriet and Heiko Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Inf.*, 26(1/2):131–192, 1988.

7 Takeshi Hayashi. On derivation trees of indexed grammars – an extension of the uvwxy-theorem. *Publ. RIMS, Kyoto Univ.*, pages 61–92, 1973.

8 Alexander Kartzow. Personal communication, via Pawel Parys, 2013.

9 Teodor Knapik, Damian Niwinski, and Pawel Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *TLCA 2001*, volume 2044 of *LNCS*, pages 253–267. Springer, 2001.

10 Naoki Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *Proc. of POPL*, pages 416–428. ACM Press, 2009.

11 Naoki Kobayashi. Model checking higher-order programs. *Journal of the ACM*, 60(3), 2013.

12 Naoki Kobayashi. Pumping by typing. In *Proceedings of LICS 2013*, pages 398–407. IEEE Computer Society, 2013.

13 Naoki Kobayashi, Kazuhiro Inaba, and Takeshi Tsukada. Unsafe order-2 tree languages are context-sensitive. In *Proceedings of FoSSaCS 2014*, volume 8412 of *LNCS*, pages 149–163. Springer, 2014.

14 Naoki Kobayashi, Kazutaka Matsuda, Ayumi Shinohara, and Kazuya Yaguchi. Functional programs as compressed data. *Higher-Order and Symbolic Computation*, 2013.

15 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of LICS 2009*, pages 179–188. IEEE Computer Society Press, 2009.

16 Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Inf. Comput.*, 243:205–221, 2015.

17 J. B. Kruskal. Well-quasi-ordering, the tree theorem, and vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225, 1960. URL: `http://www.jstor.org/stable/1993287`.

18 A. N. Maslov. The hierarchy of indexed languages of an arbitrary level. *Soviet Math. Dokl.*, 15:1170–1174, 1974.

19 C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Mathematical Proceedings of the Cambridge Philosophical Society*, 59(4):833–835, 1963.

20 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS 2006*, pages 81–90. IEEE Computer Society Press, 2006.

21 Pawel Parys. A pumping lemma for pushdown graphs of any level. In *Proceedings of STACS 2012*, volume 14 of *LIPIcs*, pages 54–65. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.

22 Pawel Parys. How many numbers can a lambda-term contain? In *Proceedings of FLOPS 2014*, volume 8475 of *LNCS*, pages 302–318. Springer, 2014. `doi:10.1007/978-3-319-07151-0_19`.

23 Pawel Parys. Intersection types and counting. *CoRR*, abs/1701.05303, 2017. A shorter version will appear in Post-proceedings of ITRS 2016. URL: `http://arxiv.org/abs/1701.05303`.

24 Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. In *Proceedings of ICALP 2011*, volume 6756 of *LNCS*, pages 162–173. Springer, 2011.

25 Kazushige Terui. Semantic evaluation, intersection types and complexity of simply typed lambda calculus. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12)*, volume 15 of *LIPIcs*, pages 323–338. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.

26 Mitchell Wand. An algebraic formulation of the Chomsky hierarchy. In *Category Theory Applied to Computation and Control*, volume 25 of *LNCS*, pages 209–213. Springer, 1974.

27 Marek Zaionc. Word operation definable in the typed lambda-calculus. *Theor. Comput. Sci.*, 52:1–14, 1987. doi:10.1016/0304-3975(87)90077-6.

28 Marek Zaionc. On the "lambda"-definable tree operations. In *Algebraic Logic and Universal Algebra in Computer Science, Conference, Ames, Iowa, USA, June 1-4, 1988, Proceedings*, volume 425 of *Lecture Notes in Computer Science*, pages 279–292, 1990.

A Strategy for Dynamic Programs: Start over and Muddle Through*†

Samir Datta[1], Anish Mukherjee[2], Thomas Schwentick[3], Nils Vortmeier[4], and Thomas Zeume[5]

1 **Chennai Mathematical Institute, Chennai, India**
sdatta@cmi.ac.in
2 **Chennai Mathematical Institute, Chennai, India**
anish@cmi.ac.in
3 **TU Dortmund, Dortmund, Germany**
thomas.schwentick@tu-dortmund.de
4 **TU Dortmund, Dortmund, Germany**
nils.vortmeier@tu-dortmund.de
5 **TU Dortmund, Dortmund, Germany**
thomas.zeume@tu-dortmund.de

Abstract

A strategy for constructing dynamic programs is introduced that utilises periodic computation of auxiliary data from scratch and the ability to maintain a query for a limited number of change steps. It is established that if some program can maintain a query for $\log n$ change steps after an AC^1-computable initialisation, it can be maintained by a first-order dynamic program as well, i.e., in DynFO. As an application, it is shown that decision and optimisation problems defined by monadic second-order (MSO) and guarded second-order logic (GSO) formulas are in DynFO, if only change sequences that produce graphs of bounded treewidth are allowed. To establish this result, Feferman–Vaught-type composition theorems for MSO and GSO are established that might be useful in their own right.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases dynamic complexity, treewidth, monadic second order logic

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.98

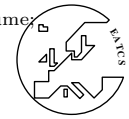
1 Introduction

Updating the result of a query after a small change to a relational database is an important problem. A theoretical framework for studying when a query can be updated in a declarative fashion was formalised by Patnaik and Immerman [11], and Dong, Su, and Topor [5]. In their formalisation, a dynamic program has a set of logical formulas that update a query after the insertion or deletion of a tuple. The formulas may use additional auxiliary relations that, of course, need to be updated as well. The queries maintainable in this way via first-order formulas constitute the dynamic complexity class DynFO.

* The full version of this paper is available as [4], http://arxiv.org/abs/1704.07998.

† The authors acknowledge the financial support by the DAAD-DST grant "Exploration of New Frontiers in Dynamic Complexity". The first and the second authors were partially funded by a grant from Infosys foundation. The second author was partially supported by a TCS PhD fellowship. The last three authors acknowledge the financial support by DFG grant SCHW 678/6-2 on "Dynamic Expressiveness of Logics".

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 98; pp. 98:1–98:14



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

Recent work has confirmed that DynFO is quite a powerful class, since it captures, e.g., the reachability query for directed graphs [2], and can even take care of pretty complex change operations [12].

In this paper, we introduce a general strategy for dynamic programs that further underscores the expressive power of DynFO. For a complexity class $\mathcal{C}$ and a function f, we call a query $\mathcal{Q}$ $(\mathcal{C}, f)$-maintainable, if there is a dynamic program (with first-order definable updates) that, starting from some input structure $\mathcal{A}$ and auxiliary relations computed in $\mathcal{C}$ from $\mathcal{A}$, can answer $\mathcal{Q}$ for $f(|\mathcal{A}|)$ many steps, where $|\mathcal{A}|$ denotes the size of the universe of $\mathcal{A}$.

We feel that this notion might be interesting in its own right. However, in this paper we concentrate on the case where $\mathcal{C}$ is (uniform) AC^1 and $f(n) = \log n$. We show that $(\mathrm{AC}^1, \log n)$-maintainable queries are actually in DynFO. We apply this insight to show that all queries and optimisation problems definable in monadic second-order logic (MSO) are in DynFO for (classes of) structures of bounded treewidth, by proving that they are $(\mathrm{AC}^1, \log n)$-maintainable. Likewise for guarded second-order logic (GSO). This implies that decision problems like 3-Colourability or HamiltonCycle as well as optimisation problems like VertexCover and DominatingSet are in DynFO, for such classes of structures.

The proof that MSO-definable queries are $(\mathrm{AC}^1, \log n)$-maintainable on structures of bounded treewidth makes use of a Feferman–Vaught-type composition theorem for MSO which might be useful in its own right.

The result that $(\mathrm{AC}^1, \log n)$-maintainable queries are in DynFO comes with a technical restriction: in a nutshell, it holds for queries that are invariant under insertion of (many) isolated elements. We call such queries *almost domain-independent* and refer to Section 3 for a precise definition.

We emphasise that the main technical challenge in maintaining MSO-queries on graphs of bounded treewidth is that tree decompositions might change drastically after an edge insertion, and can therefore not be maintained incrementally in any obvious way. In particular, the result does not simply follow from the DynFO-maintainability of regular tree languages shown in [8]. We circumvent this problem by periodically recomputing a new tree decomposition (in logarithmic space and thus in AC^1) and by showing that MSO-queries can be maintained for $\mathcal{O}(\log n)$ many change operations, even if they make the tree decomposition invalid.

Contributions

- We introduce the notion of $(\mathcal{C}, f)$-maintainability.
- We show that (almost domain-independent) $(\mathrm{AC}^1, \log n)$-maintainable queries are in DynFO.
- We show that MSO-definable (Boolean) queries are $(\mathrm{AC}^1, \log n)$-maintainable and therefore in DynFO, for structures of bounded treewidth. Likewise for MSO-definable optimisation problems and GSO-definable queries and optimisation problems.
- We state a Feferman–Vaught-type composition theorem for MSO-logic.

Related work. The simulation-based technique for proving that $(\mathrm{AC}^1, \log n)$-maintainable queries are in DynFO is inspired by proof techniques from [2] and [12]. As mentioned above, in [8] it has been shown that tree languages, i.e. MSO on trees, can be maintained in DynFO. In [1], the maintenance of parity games has been studied for graphs of bounded treewidth, though in the restricted setting where the tree decomposition stays the same for all changes.

Organisation. Basic terminology is recalled in Section 2, followed by a short introduction into dynamic complexity in Section 3. In Section 4 we introduce the notion of $(\mathcal{C}, f)$-maintainability and show that $(\mathrm{AC}^1, \log n)$-maintainable queries are in DynFO. A glimpse on the proof techniques for proving that MSO and GSO queries are in DynFO for graphs of bounded treewidth is given in Section 5 via the example 3-Colourability. The proof of the general results is presented in Section 6. An extension to optimisation problems can be found in Section 7. For many proofs, details are deferred to the full version of this paper [4].

2 Preliminaries

We assume familiarity with first-order logic FO and other notions from finite model theory [9, 10]. Some further notation regarding MSO logic and types will be introduced in Section 6.

In this paper we consider finite relational structures over relational signatures $\Sigma = \{R_1, \ldots, R_\ell, c_1, \ldots, c_m\}$, where each R_i is a relation symbol with a corresponding arity $\mathrm{Ar}(R_i)$, and each c_j is a constant symbol. A Σ-structure $\mathcal{A}$ consists of a finite *domain* A, a relation $R_i^{\mathcal{A}} \subseteq A^{\mathrm{Ar}(R_i)}$, and a constant $c_j^{\mathcal{A}} \in A$, for each $i \in \{1, \ldots, \ell\}, j \in \{1, \ldots, m\}$. Sometimes, especially in Section 3, we consider relational structures as relational databases This is basically a different terminology that is common in the context of dynamic complexity, since the original motivation for considering the class DynFO came from relational databases. In particular, the class DynFO will be defined as a class of queries of arbitrary arity.

However, we will mostly consider Boolean queries over structures with a single binary relation symbol E, which can equivalently be viewed as decision problems for graphs $G = (V, E)$. For a set $U \subseteq V$, $G[U]$ denotes the induced subgraph $(U, E \cap (U \times U))$.

We will often use structures that have a linear order $\leq$ and compatible ternary relations encoding arithmetical operations $+$ and $\times$ or a binary BIT relation on the universe. We write $\mathrm{FO}(+, \times)$ or $\mathrm{FO}(\mathrm{BIT})$ to emphasise that we allow first-order formulas to use such additional relations.[1] We also use that $\mathrm{FO}(+, \times) = \mathrm{FO}(\mathrm{BIT})$ [9].

A *tree decomposition* (T, B) of G consists of a (rooted) tree $T = (I, F, r)$ and a function $B \colon I \to 2^V$ such that (1) for all $v \in V$, the set $\{i \in I \mid v \in B(i)\}$ is non-empty, (2) for all $(u, v) \in E$, there is an $i \in I$ with $\{u, v\} \subseteq B(i)$, and (3) the subgraph $T[\{i \in I \mid v \in B(i)\}]$ is connected. We refer to the number of children of a node of T as its *degree*. We denote the parent node of a node i by $p(i)$. The *width* of a tree decomposition is defined as the maximal size of a bag minus 1. The *treewidth* of a graph G is the minimal width among all tree decompositions of G. A tree decomposition is *nice* if (1) T has depth at most $\mathcal{O}(\log n)$, (2) the degree of the nodes is at most 2, and (3) all bags are distinct. We use the following lemma which is an adaption of [7, Lemma 3.1].

▶ **Lemma 1.** *For every graph of treewidth k, a nice tree decomposition of width $4k + 5$ can be computed in logarithmic space.*

In this paper we only consider nice tree decompositions, and due to property (3) of these decompositions we can identify bags with nodes from I.

For two nodes i, i' of I, we write $i' \preceq i$ if i' is in the subtree of T rooted at i and $i' \prec i$ if, in addition, $i' \neq i$. A *triangle* δ of T is a triple (i_0, i_1, i_2) of nodes from I such that $i_1 \preceq i_0$, $i_2 \preceq i_0$, and (1) $i_1 = i_2$ or (2) neither $i_1 \preceq i_2$ nor $i_2 \preceq i_1$. In case of (2) we call the triangle *proper*, in case of (1) *unary*, unless $i_0 = i_1 = i_2$ in which we call it *open*.

[1] The question of $<$-invariance will not be relevant in the context of this paper since the order of insertion of elements to a structure will determine a linear order on the universe.

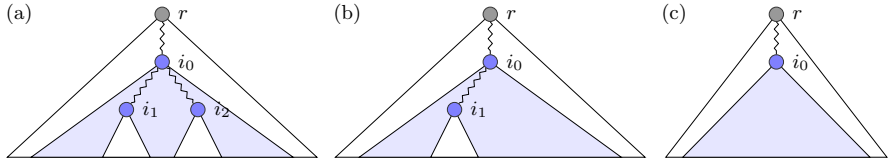


Figure 1 Illustration of (a) a proper triangle (i_0, i_1, i_2), (b) a unary triangle (i_0, i_1, i_1), and (c) an open triangle (i_1, i_1, i_1). The blue shaded area is the part of the tree contained in the triangle.

The subtree $T(\delta)$ *induced* by a triangle consists of all nodes j of T for which the following holds: (i) $j \preceq i_0$, (ii) if $i_1 \prec i_0$ then $j \not\preceq i_1$, and (iii) if $i_2 \prec i_0$ then $j \not\preceq i_2$. That is, for a proper or unary triangle, $T(\delta)$ contains all nodes of the subtree rooted at i_0 which are not below i_1 or i_2. For an open triangle $\delta = (i_0, i_0, i_0)$, $T(\delta)$ is just the subtree rooted at i_0.

Each triangle δ induces a subgraph $G(\delta)$ of G as follows: $V(\delta)$ is the union of all bags of $T(\delta)$. By $B(\delta)$ we denote the set $B(i_0) \cup B(i_1) \cup B(i_2)$ of *interface nodes* of $V(\delta)$. All other nodes are called *inner nodes*. The edge set of $G(\delta)$ consists of all edges of G that involve at least one inner node of $V(\delta)$.

Our main result refers to the complexity class (uniform) AC^1 whose definition can be found, e.g., in [15]. The precise definition of the class is not relevant for this paper. It suffices to know that it contains the classes LOGSPACE and NL and that it can be characterised as the class $\mathrm{IND}[\log n]$ of problems that can be expressed by applying a first-order formula $\mathcal{O}(\log n)$ times [9, Theorem 5.22]. Here, n denotes the size of the universe and the formulas can use built-in relations $+$ and $\times$. Our proofs often assume that $\log n$ is a natural number, but they can be easily adapted to the general case.

3 Dynamic Complexity

We briefly repeat the essentials of dynamic complexity, closely following [13, 3].

The goal of a dynamic program is to answer a given query on an *input database* subjected to changes that insert or delete single tuples. The program may use an auxiliary data structure represented by an *auxiliary database* over the same domain. Initially, both input and auxiliary database are empty; and the domain is fixed during each run of the program.

A dynamic program has a set of update rules that specify how auxiliary relations are updated after a change of the input database. An *update rule* for updating an auxiliary relation T is basically a formula φ. As an example, if $\varphi(\vec{x}, \vec{y})$ is the update rule for auxiliary relation T under insertions into input relation R, then the new version of T after insertion of a tuple $\bar{a}$ to R is $T \stackrel{\text{def}}{=} \{\vec{b} \mid (\mathcal{I}, Aux) \models \varphi(\vec{a}, \vec{b})\}$ where $\mathcal{I}$ and Aux are the current input and auxiliary databases. For a state $\mathcal{S} = (\mathcal{I}, Aux)$ of the dynamic program with input database $\mathcal{I}$ and auxiliary database Aux we denote the state of the program after applying the change sequence α by $\mathcal{P}_\alpha(\mathcal{S})$. The dynamic program *maintains* a k-ary query $\mathcal{Q}$ if, for each non-empty sequence α of changes and each empty input structure $\mathcal{I}_\emptyset$, relation Q in $\mathcal{P}_\alpha(\mathcal{S}_\emptyset)$ and $\mathcal{Q}(\alpha(\mathcal{I}_\emptyset))$ coincide. Here, $\mathcal{S}_\emptyset = (\mathcal{I}_\emptyset, Aux_\emptyset)$, where $Aux_\emptyset$ denotes the empty auxiliary structure over the domain of $\mathcal{I}_\emptyset$, and $\alpha(\mathcal{I}_\emptyset)$ is the input database after applying α.

In this paper, we are particularly interested in maintaining queries for structures of bounded treewidth. There are several ways to adjust the dynamic setting to restricted classes $\mathcal{C}$ of structures. Here, we simply disallow change sequences that construct structures outside $\mathcal{C}$. That is, in the above definition, only change sequences α are considered, for which each prefix transforms an initially empty structure into a structure from $\mathcal{C}$. We say that a program

maintains $\mathcal{Q}$ for a class $\mathcal{C}$ of structures, if Q contains its result after each change sequence α such that the application of each prefix of α to $\mathcal{I}_\emptyset$ yields a structure from $\mathcal{C}$.

The class of queries that can be maintained by a dynamic program is called DynFO. Programs for queries in DynFO$(+,\times)$ have three particular auxiliary relations that are initialised as a linear order and the corresponding addition and multiplication relations.

We say that a query $\mathcal{Q}$ is in DynFO for a class $\mathcal{C}$ of structures, if there is a dynamic program that maintains $\mathcal{Q}$ for $\mathcal{C}$.

The *active domain* $\mathrm{adom}(\mathcal{A})$ of a structure $\mathcal{A}$ contains all elements used in some tuple of $\mathcal{A}$. A query $\mathcal{Q}$ is *almost domain-independent* if there is a $c \in \mathbb{N}$ such that $\mathcal{Q}(\mathcal{A}) \restriction (\mathrm{adom}(\mathcal{A}) \cup B) = \mathcal{Q}(\mathcal{A} \restriction (\mathrm{adom}(\mathcal{A}) \cup B))$ for all structures $\mathcal{A}$ and sets $B \subseteq A \setminus \mathrm{adom}(\mathcal{A})$ with $|B| \geq c$. The following proposition adapts Proposition 7 from [3].

▶ **Proposition 2.** *If a query $\mathcal{Q} \in$ DynFO$(+,\times)$ is almost domain-independent, then also $\mathcal{Q} \in$ DynFO.*

4 Algorithmic Technique

There are alternative definitions of DynFO, where the initial structure is non-empty and the initial auxiliary relations can be computed within some complexity [11, 16]. However, in a practical scenario of dynamic query answering it is conceivable that the quality of the auxiliary relations decreases over time and that they are therefore recomputed from scratch at times. We formalise this notion by a relaxed definition of maintainability in which the initial structure is non-empty, the dynamic program is allowed to apply some preprocessing, and query answers need only be given for a certain number of change steps.

We call a query $\mathcal{Q}$ $(\mathcal{C}, f)$-*maintainable*, for some complexity class[2] $\mathcal{C}$ and some function $f : \mathbb{N} \to \mathbb{R}$, if there is a dynamic program $\mathcal{P}$ and a $\mathcal{C}$-algorithm $\mathcal{A}$ such that for each input database $\mathcal{I}$ over a domain of size n, each linear order $\leq$ on the domain, and each change sequence α of length $|\alpha| \leq f(n)$, the relation Q in $\mathcal{P}_\alpha(\mathcal{S})$ and $\mathcal{Q}(\alpha(\mathcal{I}))$ coincide where $\mathcal{S} = (\mathcal{I}, \mathcal{A}(\mathcal{I}, \leq))$.

Although we feel that $(\mathcal{C}, f)$-maintainability deserves further investigation, in this paper we exclusively use it as a tool to prove that queries are actually maintainable in DynFO. To this end, we show next that every $(\mathrm{AC}^1, \log n)$-maintainable query is actually in DynFO and prove later that the queries in which we are interested are $(\mathrm{AC}^1, \log n)$-maintainable.

▶ **Theorem 3.** *Every $(\mathrm{AC}^1, \log n)$-maintainable, almost domain-independent query is in* DynFO.

Proof Sketch (of Theorem 3. Assume that a dynamic program $\mathcal{P}$ witnesses that an almost domain-independent query $\mathcal{Q}$ is $(\mathrm{AC}^1, \log n)$-maintainable. Thanks to Proposition 2 it suffices to construct a dynamic program $\mathcal{P}'$ that witnesses $\mathcal{Q} \in$ DynFO$(+,\times)$. We restrict ourselves to graphs, for simplicity.

The overall idea is to use a simulation technique similar to the ones used in [2] and [12]. We consider each application of one change as a *time step*. We refer to the graph after time step t as $G_t = (V, E_t)$. After each time step t, $\mathcal{P}'$ starts a thread that uses $\frac{1}{2} \log n$ steps to compute the auxiliary relations for G_t (using $\mathrm{AC}^1 = \mathrm{IND}[\log n]$) and then another $\frac{1}{2} \log n$ steps to apply the $\log n$ changes of time steps $t+1, \ldots, t+\log n$ (two at a time). After these

[2] Strictly speaking $\mathcal{C}$ should be a complexity class of functions. In this paper, the implied class of functions will always be clear from the stated class of decision problems.

$\log n$ steps the thread is ready to answer query $\mathcal{Q}$ about $G_{t+\log n}$ at time step $t+\log n$. Since one such thread starts at every time point, the program can answer query $\mathcal{Q}$, for each time point $\geq \log n$.

We next give details on the two phases and describe how to deal with earlier time points.

For the first phase, we make use of the equality $\mathrm{AC}^1 = \mathrm{IND}[\log n]$. Let ψ be an inductive formula that is applied $d \log n$ times, for some d, to get the auxiliary relations for a given graph G and the given order $\leq$. The program $\mathcal{P}'$ simply applies ψ to G_t for $2d$ times during each time step, and thus the fixpoint of ψ is reached after $\frac{1}{2}\log n$ steps. The change operations that occur during these steps are not applied to G_t directly but rather stored in some additional relation.

During the second phase the $\frac{1}{2}\log n$ stored change operations and the $\frac{1}{2}\log n$ change operations that happen during the next $\frac{1}{2}\log n$ steps are applied to the state after phase 1. To this end, it suffices for $\mathcal{P}'$ to apply two changes during each time step by simulating two update steps of $\mathcal{P}$. Since $\mathcal{P}$ can maintain $\mathcal{Q}$ for $\log n$ changes, at the end of phase 2, at time point $t+\log n$, $\mathcal{P}'$ can give the correct query answer for $\mathcal{Q}$ about $G_{t+\log n}$.

To enable $\mathcal{P}'$ to answer $\mathcal{Q}$ also for time steps $t < \log n$, it proceeds as follows. It starts a new thread at time $\frac{t}{2}$ with a graph with at most $\frac{t}{2}$ edges and applies ψ relative to a domain D_t of size $2t+c$, where c is the constant from (almost) domain-independence. The first phase of this thread lasts from time points $\frac{t}{2}+1$ to $\frac{3t}{4}$, and applies ψ for $4d$ times during each step. As a fixpoint is reached after $\frac{\log(2t+c)}{4} < \frac{t}{4}$ steps, the auxiliary relations are initialised properly (very small t can be handled separately). From time $\frac{3t}{4}+1$ to time t the changes are applied, again two at a time and the thread is ready to answer $\mathcal{Q}$ at time point t. As at time t at most $2t$ elements are used by edges, the almost domain-independence of $\mathcal{Q}$ guarantees that the result computed by the thread relative to D_t coincides with the D_t-restriction of the query result for G_t. The full query result for G_t, possibly including tuples with elements from $V \setminus D_t$, is obtained as follows: a tuple $\bar{t}$ is included in the query result, if it can be generated from a tuple $\bar{t}'$ of the restricted query result by replacing elements from $D_t \setminus \mathrm{adom}(G_t)$ by elements from $V \setminus D_t$ (under consideration of equality constraints among these elements).

The above presentation assumes a separate thread for each time point and each thread uses its own relations. These threads can be combined into one dynamic program as follows. We can safely assume that $n \geq \log n$ and since at each time point at most $\log n$ threads are active, we can number them in a round robin fashion with numbers $1, \ldots, n$. The arity of all auxiliary relations is incremented by one and the additional dimension is used to indicate the number of the thread to which a tuple belongs. ◀

5 Warm-up: 3-Colourability

In this section, we show that the 3-colourability problem 3Col for graphs of bounded treewidth can be maintained in DynFO. Given an undirected graph, 3Col asks whether its vertices can be coloured with three colours such that adjacent vertices have different colours.

▶ **Theorem 4.** *For every k,* 3Col *is in* DynFO *for graphs with treewidth at most k.*

The remainder of this section is dedicated to a proof sketch for this theorem. Thanks to Theorem 3 and the fact that 3Col is almost domain-independent, it suffices to show that 3Col is $(\mathrm{AC}^1, \log n)$-maintainable for graphs with treewidth at most k. In a nutshell, our approach can be summarised as follows.

The AC^1-initialisation computes a nice tree decomposition $T = (I, F, r)$ of width at most $4k+5$ and maximum bag size $\ell \stackrel{\text{def}}{=} 4k+6$, as well as information about the 3-colourability

of induced subgraphs of G. More precisely, it computes, for each triangle δ of T and each 3-colouring C of the nodes of $B(\delta)$, whether there exists a colouring C' of the inner vertices of $G(\delta)$, such that all edges involving at least one inner vertex are consistent with $C \cup C'$.

During the following $\log n$ change operations, the dynamic program does not need to do much. It only maintains a set S of *special* bags: for each *affected* graph node v that participates in any changed (i.e. deleted or inserted) edge, S contains one bag in which v occurs. Also, if two bags are special, their least common ancestor is considered special and is included in S. It will be guaranteed that there are at most $4 \log n$ special bags. With the auxiliary information, a first-order formula φ can test whether G is 3-colourable as follows. By existentially quantifying 8ℓ variables, the formula can choose two bits of information for each of the at most $4\ell \log n$ nodes in special bags. For each such node, these two bits are interpreted as encoding of one of three colours and all that the formula φ needs to do is checking that this colouring of the special bags can be extended to a colouring of G. This can be done with the help of the auxiliary relations computed during the initialisation which provide all necessary information about subgraphs induced by triangles consisting of special bags.

6 MSO and GSO Queries

In this section, we show that for each k and each MSO-sentence φ the model checking problem for φ on structures of treewidth at most k is in DynFO.

After some definitions regarding MSO types, we will state a Feferman–Vaught-type composition theorem for the composition of at most $\mathcal{O}(\log n)$ many structures that meet in a set C of at most $\mathcal{O}(\log n)$ elements. We will show that if the structure is suitably extended by information about the types of the (disjoint) structures outside C, then MSO formulas can be replaced by first-order formulas. This part is formulated for arbitrary relational structures instead of graphs since we think it might be useful in other contexts as well.

Afterwards, we will use the Feferman–Vaught-type composition theorem to show the maintainability of MSO properties on structures of bounded treewidth. Finally, we explain how these results can be lifted to guarded second-order logic.

6.1 MSO-types

MSO-logic is the extension of first-order logic, which allows existential and universal quantification over set variables $X, X_1, \ldots$. The depth of a MSO formula is the maximum nesting depth of (second-order and first-order) quantifiers in the syntax tree of the formula. For a signature Σ and a natural number $d \geq 0$, the *depth-d MSO-type* of a Σ-structure $\mathcal{A}$ is defined as the set of all MSO-sentences φ over Σ of quantifier depth at most d, for which $\mathcal{A} \models \varphi$ holds.

We also need to deal with situations, where we have to take a variable assignment and some additional elements of the structure into account, and therefore the general notion of types is slightly more involved. Let $\mathcal{A}$ be a Σ-structure and $\bar{v} = (v_1, \ldots, v_m)$ a tuple of elements from $\mathcal{A}$. We write $(\mathcal{A}, \bar{v})$ for the structure over $\Sigma \cup \{c_1, \ldots, c_m\}$ which interprets c_i as v_i, for every $i \in \{1, \ldots, m\}$. For a set $\mathcal{Y}$ of first-order and second-order variables and an assignment α for the variables of $\mathcal{Y}$, the depth-d MSO-type of $(\mathcal{A}, \bar{v}, \alpha)$ is the set of MSO-formulas with free variables from $\mathcal{Y}$ of depth d that hold in $(\mathcal{A}, \bar{v}, \alpha)$.

We summarise some basic properties of types in the following. Unless not otherwise stated, *type* always refers to MSO-type. For any $d' < d$ the depth-d' type of a structure results from its depth-d type by simply removing all formulas of depth larger than d'.

For every depth-d type, there is a depth-d MSO formula α_τ that is true in exactly the structures and for those assignments of type τ.

Each depth-d type τ induces a set of depth-$(d-1)$ types over $\mathcal{Y} \cup \{x\}$ (assuming $x \notin \mathcal{Y}$) that can be realised in a structure of type τ, represented by the set of all formulas $\alpha_{\tau'}$ for depth-$(d-1)$ types τ' with free variables set $\mathcal{Y} \cup \{x\}$, for which $\exists x \alpha_{\tau'}$ is in τ. We call a type τ', for which $\exists x\ \alpha_{\tau'}$ is in τ, an x-realisation of τ. Likewise, a depth-$(d-1)$ type τ' is an X-realisation for a depth-d type τ, if $\exists X\ \alpha_{\tau'}$ is in τ. For more background on MSO-logic, types, and the above properties readers might consult, e.g., [10].

6.2 A Feferman–Vaught-type composition theorem

In the following, we give an adaptation of the Feferman–Vaught-type composition theorem from [6] that will be useful for maintaining MSO properties.

Intuitively, the idea is very easy, but the formal presentation will come with some technical complications. For simplicity, we explain the basic idea for graphs first.

In a nutshell, we consider graphs $G = (V, E)$ with a *center* $C \subseteq V$, such that the graph $G[V - C]$ is a disjoint union of components $D_1 - C, \ldots, D_\ell - C$, such that, for some $w > 0$,

- $|D_i \cap C| \leq w$, for every i,
- all edges in E have both end nodes in C or in some D_i, and
- for each i there is some element $v_i \in D_i \cap C$ that is not contained in any D_j, for $j \neq i$.

In this case, we say that $(C, D_1, \ldots, D_\ell, v_1, \ldots, v_\ell)$ is a *weak partition* of G with *center* C, and *connection width* w. We refer to the sets $D_1, \ldots, D_\ell$ as *petals* and the nodes $v_1, \ldots, v_\ell$ as *identifiers* of their respective petals. We emphasise that ℓ is not assumed to be bounded by any constant, only by $|C|$.

Readers who have read the proof sketch for Theorem 4 can think of C as the set of vertices from special bags (plus one inner vertex per clean triangle as identifier).

Our goal is to show that, if a graph G with a weak partition of logarithmic center size is extended by the information about the MSO types of its petals in a suitable way, resulting in a structure G', then MSO formulas over G have equivalent first-order formulas over G'.

In a first step, we show that, if (the center) of G is suitably extended by the information about the MSO types of its petals, then every MSO formula has an equivalent MSO formula whose quantification is restricted to C.[3] In a second step we show that, if in a MSO formula quantification is restricted to some node set C of logarithmic size then there is an equivalent (unrestricted) first-order formula. For the second step we assume that the graph has an additional relation that encodes subsets of C by bounded-size tuples over V.

In the following, we work out the above plan in more detail. We fix some relational signature Σ and assume that it contains a unary relation symbol C.

The definition of weak partitions easily carries over to general Σ-structures. In particular, tuples need to be entirely in C or in some petal D_i. For every i, we call the set $I_i \stackrel{\text{def}}{=} D_i \cap C$ the *interface* of D_i and the nodes of a petal D_i that are *not* in C *inner elements* of D_i.

Let $\mathcal{A}$ be a Σ-structure, $P = (C, D_1, \ldots, D_\ell, v_1, \ldots, v_\ell)$ a weak partition of connection width w, and $d > 0$. For every i, let $\bar{u}^i = (u^i_1, \ldots, u^i_w)$ be a tuple of elements from the interface of D_i such that $u^i_1 = v_i$ and every node from I_i occurs in $\bar{u}^i$. By $\mathcal{A}_i$ we denote the substructure of $\mathcal{A}$ induced by D_i with $u^i_1, \ldots, u^i_w$ as constants but *without* all tuples over C,

[3] As remarked by a reviewer, Proposition 5 below can probably be concluded from Shelah's Composition Theorem for generalised sums [14].

i.e., $\mathcal{A}_i$ only contains tuples with at least one inner element of D_i. The *depth d, width w MSO indicator structure* of $\mathcal{A}$ relative to P and tuples $\bar{u}^i$ is the unique structure $\mathcal{B}$ such that:

- $\mathcal{B}$ is an expansion of $\mathcal{A}$ (with the same universe and the same Σ-relations),
- $\mathcal{B}$ has an additional w-ary relation J that contains all tuples $\bar{u}^i$, and
- $\mathcal{B}$ has additional unary relation symbols R_τ, one for every depth-d MSO-type over $\Sigma \cup \{c_1, \ldots, c_w\}$, and for each such τ, R_τ contains the identifier nodes v_i, for all i, for which the depth-d MSO-type of $(\mathcal{A}_i, \bar{u}^i)$ is τ.

The set of all indicator structures of $\mathcal{A}$ relative to P for varying tuples $\bar{u}^i$ is denoted by $\mathcal{S}(\mathcal{A}, P, w, d)$.

We call a MSO-formula *C-restricted*, if all its quantified subformulas are of one of the following forms.

- $\exists x\ (C(x) \wedge \varphi)$ or $\forall x\ (C(x) \to \varphi)$,
- $\exists X\ (\forall x(X(x) \to C(x)) \wedge \varphi)$ or $\forall X\ (\forall x(X(x) \to C(x)) \to \varphi)$.

▶ **Proposition 5.** *For each $d > 0$, every MSO sentence φ with depth d, and each w, there is a C-restricted MSO sentence ψ such that for every Σ-structure $\mathcal{A}$ with a weak partition P of connection width w and every $\mathcal{B} \in \mathcal{S}(\mathcal{A}, P, w, d)$ it holds $\mathcal{A} \models \varphi$ if and only if $\mathcal{B} \models \psi$.*

Proof. The construction of ψ and the proof of its correctness is by induction on the structure of φ. It can be found in the full version of the paper [4]. ◀

To formalise the second step, we need some further notation. Let $\mathcal{A}$ be a structure with a unary relation C and a $(k+1)$-ary relation Sub, for some k. We say that Sub *encodes subsets of C* if, for each subset $C' \subseteq C$, there is a k-tuple $\bar{t}$ such that, for every element $c \in C$ it holds $c \in C'$ if and only if $(\bar{t}, c) \in \text{Sub}$. Clearly, such an encoding of subsets only exists if $|V|^k \geq 2^{|C|}$ and thus if $|C| \leq k \log |V|$.

▶ **Proposition 6.** *For each C-restricted MSO-sentence ψ over a signature Σ (containing C) and every k there is a first-order sentence χ over $\Sigma \cup \{S\}$ where S is a $(k+1)$-ary relation symbol such that, for every Σ-structure $\mathcal{A}$ and $(k+1)$-ary relation Sub that encodes subsets of C (in $\mathcal{A}$), it holds $\mathcal{A} \models \psi$ if and only if $(\mathcal{A}, \mathit{Sub}) \models \chi$.*

Proof. The proof is straightforward. Formulas $\exists X\ (\forall x(X(x) \to C(x)) \wedge \varphi)$ are translated into formulas $\exists \bar{x}\ \varphi'$, where $\bar{x}$ is a tuple of k variables and φ' results from φ by simply replacing every atomic formula $X(y)$ by $\text{Sub}(\bar{x}, y)$. And likewise for universal set quantification. ◀

By combining Propositions 5 and 6 we immediately get the following result.

▶ **Theorem 7.** *For each $d > 0$, every MSO sentence φ with depth d, and each w, there is a first-order sentence χ such that, for every Σ-structure $\mathcal{A}$ with a weak partition $P = (C, D_1, \ldots, D_\ell, v_1, \ldots, v_\ell)$ of connection width w, every $\mathcal{B} \in \mathcal{S}(\mathcal{A}, P, w, d)$ and a relation Sub that encodes subsets of C, it holds $\mathcal{A} \models \varphi$ if and only if $(\mathcal{B}, \mathit{Sub}) \models \chi$.*

6.3 MSO on structures of bounded treewidth

In this subsection we prove a dynamic version of Courcelle's Theorem: all MSO properties can be maintained in DynFO for graphs with bounded treewidth. More precisely, for a given MSO sentence φ we consider the model checking problem MC_φ that asks whether a given graph G satisfies φ, that is, whether $G \models \varphi$ holds.

▶ **Theorem 8.** *For every* MSO *sentence φ and every k, MC_φ is in* DynFO *for graphs with treewidth at most k.*

Thanks to Theorem 3 it suffices to show that MC_φ is $(\mathrm{AC}^1, \log n)$-maintainable for graphs G with treewidth at most k. We note that it is easy to see MSO-definable queries are almost domain-independent and the respective constant c depends only on the formula φ. The reason is that MSO-formulas can not make use of more than a constant number of isolated nodes.[4] The dynamic program that will be constructed in the proof works very similarly to the one of Theorem 4: during its initialisation it constructs a tree decomposition and appropriate MSO-types for all triangles. During the change sequence, a set C of special nodes is used that contains, for each affected graph node v, at least one bag containing v. The union of all bags represented by the set C induces a weak partition P and the dynamic program basically maintains an MSO indicator structure for G relative to P. Since there are only $\mathcal{O}(\log n)$ many change steps, $|C| = \mathcal{O}(\log n)$ and therefore Theorem 7 yields that from this auxiliary data it can be inferred in a first-order fashion whether $G \models \varphi$.

Proof (of Theorem 8). Thanks to Theorem 3 and the fact that MSO queries are almost domain-independent it suffices to show that MC_φ is $(\mathrm{AC}^1, \log n)$-maintainable in DynFO for graphs with treewidth at most k. Let d be the quantifier depth of φ.

Given a graph $G = (V, E)$, the AC^1 initialisation first computes a nice tree decomposition $T = (I, F, r)$ with bags of size at most $\ell \stackrel{\text{def}}{=} 4k + 6$, together with $\preceq$. With each node i, we associate a tuple $\bar{v}(i) = (v_1, \ldots, v_m, v_1, \ldots, v_1)$ of length ℓ, where $B(i) = \{v_1, \ldots, v_m\}$ and $v_1 < \cdots < v_m$. That is, if the bag size of i is ℓ, this tuple just contains all graph nodes of the bag in increasing order. If the bag size is smaller, the smallest graph node is repeated. The AC^1-initialisation also ensures that arithmetic relations $+$, $\times$ and BIT are available.

The dynamic program further uses additional auxiliary relations S, N, and D_τ, for each depth-d MSO-type τ over the signature that consists of the binary relation symbol E and $3\ell + 1$ constant symbols $c_1, \ldots, c_{3\ell+1}$. From these relations all ingredients needed to apply Theorem 7 can be first-order defined: a weak partition P with center C, an MSO indicator structure, and a relation Sub that encodes subsets of C.

The relation S stores tuples representing special bags, as in the proof of Theorem 4. The relations D_τ provide MSO type information for all triangles. More precisely, for each triangle $\delta = (i_0, i_1, i_2)$ for which the subgraph $G(\delta)$ has at least one inner node, D_τ contains the tuple $(v(\delta), \bar{v}(i_0), \bar{v}(i_1), \bar{v}(i_2))$ if and only if the MSO depth-d type of $(G(\delta), v(\delta), \bar{v}(i_0), \bar{v}(i_1), \bar{v}(i_2))$ is τ, where $v(\delta)$ denotes the smallest inner node of $G(\delta)$ with respect to $\leq$.

The set C always contains all graph nodes that occur in special bags (and thus in S), plus one inner node $v(\delta)$, for each maximal[5] clean triangle with at least one inner node. Relation N maintains a bijection between C and an initial segment of $\leq$. From the auxiliary relations, the relations used for Theorem 7 can be defined as follows.

The relations S and C are used to define a weak partition as follows. Clean triangles with at least two inner nodes become petals of the weak partition. Thus the interface $I(\delta)$ of a petal corresponding to a clean triangle $\delta = (i_0, i_1, i_2)$ contains the nodes from $B(i_0), B(i_1)$, and $B(i_2)$ as well as the node $v(\delta)$. Now, an indicator structure $\mathcal{B} \in \mathcal{S}(\mathcal{A}, P, w, d)$ can be first-order defined as follows. Clearly, clean triangles can be easily first-order defined from the relation S. For each clean triangle $\delta = (i_0, i_1, i_2)$ with at least two inner nodes, the relation J contains a tuple $(\delta(v), \bar{v}(i_0), \bar{v}(i_1), \bar{v}(i_2))$, and the relation R_τ contains $\delta(v)$ if and only if is $(\delta(v), \bar{v}(i_0), \bar{v}(i_1), \bar{v}(i_2)) \in D_\tau$. For defining Sub, we observe that C is of size $b \log n$ for some $b \in \mathbb{N}$. Thus a subset C' of C can be represented by a tuple $(a_1, \ldots, a_b)$ of nodes,

[4] It should be mentioned that structures with a linear order $\leq$ do not have any isolated nodes.
[5] Maximal basically means that all its corner nodes are special.

where an element $c \in C$ is in C' if and only if c is the m-th element of C with respect to the mapping defined by N, $m = (\ell - 1)\log n + j$ and the j-th bit of a_ℓ is one. It is easy to see that the relations can be first-order defined from S, N and D_τ.

How the auxiliary relations are initialised and updated is detailed in the full version. ◀

6.4 Extension to GSO logic

We finally sketch how the results of this section can be extended to guarded second-order logic (GSO). In a nutshell, GSO extends MSO by *guarded second-order quantification*. Thus, it syntactically allows to quantify over non-unary relation variables. However, this quantification is semantically restricted: a tuple $\bar{t} = (a_1, \ldots, a_m)$ can only occur in a quantified relation, if all elements from $\{a_1, \ldots, a_m\}$ occur together in some tuple of the structure, in which the formula is evaluated.

To state the analogue of Proposition 5 for GSO, two definitions need to be modified: GSO indicator structures store information about the respective GSO types instead of MSO types. C-restricted formulas can use GSO-quantifiers only to quantify relations over C, e.g., formulas need to be restricted as in $\exists X\ (\forall \bar{x}(X(x_1, \ldots, x_m) \rightarrow (C(x_1) \wedge \cdots \wedge C(x_m))) \wedge \varphi)$. In the statement of Proposition 5 MSO can simply be replaced by GSO. The proof hardly changes. Of course, there is an additional case for GSO quantification but the types of petals can still be handled by MSO quantification. For Proposition 6, encoding of subsets has to be extended to encoding of subrelations. For the quantification of m-ary relations this encoding has to be done by a $(k+m)$-ary relation, for some k. Such an encoding only exists, if the number of tuples over C in $\mathcal{A}$ is only logarithmic. Analogously, Theorem 7 can be extended.

7 MSO Optimisation Problems

With the techniques presented in the previous section also MSO definable optimisation problems can be maintained in DynFO for graphs with bounded treewidth. An MSO definable optimisation problem OPT_φ is induced by an MSO formula $\varphi(X)$ with a free set variable X. Given a graph G with vertex set V, it asks for a set $A \subseteq V$ of minimal[6] size such that $G \models \varphi(A)$.

From the point of view of dynamic programs, such an optimisation problem is just a unary query, that is, the result is defined by some formula $\psi(x)$ with a free first-order variable x.

▶ **Theorem 9.** *For every* MSO *formula* $\varphi(X)$ *and every* k, OPT_φ *is in* DynFO *for graphs with treewidth at most* k.

A dynamic program for an optimisation problem OPT_φ can be constructed by a modification of a program for the decision problem for the MSO sentence $\exists X\ \varphi$, as constructed in the proof of Theorem 8. Basically, we enrich the type information for each petal by information about the smallest set with which a given (X-realisation) type τ can be obtained.

Proof Sketch (of Theorem 9). We describe how a dynamic program for model checking $\psi \stackrel{\text{def}}{=} \exists X\ \varphi$, where $\varphi(X)$ is an MSO-formula of quantifier depth d, can be adapted to a program for OPT_φ. We reuse the notation from the proof of Theorem 8.

Most auxiliary relations remain as in the program of that proof. However, instead of relations D_τ for depth-$(d+1)$-types τ the program uses relations $\#D_{\tau'}$ and $S_{\tau'}$ for

[6] The adaptation to maximisation problems is straightforward.

X-realisations τ' of such types with the following intention. Let δ be a triangle. If τ' is a depth-d type over $\{E, X, c_1, \ldots, c_{3\ell+1}\}$ that can be realised by some set A, then for the minimal size s of such a set A, $\#D_{\tau'}$ shall contain a tuple $(v(\delta), \bar{v}(i_0), \bar{v}(i_1), \bar{v}(i_2), v_s)$, similarly as in the proof of Theorem 8, but with v_s chosen as the $(s+1)$-th element[7] with respect to $\leq$. Furthermore, for the lexicographically minimal set A of this kind and size s, $S_{\tau'}$ shall contain all tuples $(v(\delta), \bar{v}(i_0), \bar{v}(i_1), \bar{v}(i_2), v)$, where $v \in A$.

The proof of Theorem 8 can be extended to show that the initial versions of these auxiliary relations can be computed in AC^1. For the inductive step of this computation, a type τ of a triangle δ might be achievable by a finite number of combinations of types of its sub-triangles. Here, the overall sizes of the underlying sets for X need to be computed and the minimal solution needs to be picked. This is possible by a $\mathrm{FO}(+,\times)$-formula since the number of possible combinations is bounded by a constant depending only on d and k.

The updates of the auxiliary relations are exactly as in the proof of Theorem 8. Since D_τ needs no updates there, neither $\#D_\tau$ nor S_τ do, here.

It remains to explain how the actual query result can be defined. A close inspection of the proofs of Propositions 5 and 6 reveals that the first-order formula χ equivalent to $\exists X \varphi$, as guaranteed by Theorem 7, is of the form $\exists \bar{x} \exists (\bar{x}_\tau)_\tau \hat{\chi}$, where, in a nutshell, $\bar{x}$ represents all center nodes from C in X and $\bar{x}_\tau$ selects all petals $G(\delta_j)$ that have depth-d MSO type τ. The formula χ can first be adapted such that it defines the set of all nodes v selected in C or occurring in a tuple $(v(\delta), \bar{v}(i_0), \bar{v}(i_1), \bar{v}(i_2), v)$ of $S_{\tau'}$ for the type τ' that was chosen for the petal associated with δ. Next, using the ability of $\mathrm{FO}(+,\times)$ to add up logarithmically many numbers [15, Theorem 1.21], the size of the thus represented set can be determined using $\#D_{\tau'}$. Then, a similar formula can check that there is no (lexicographically) smaller set that makes φ true. We emphasise that the resulting formula so far can become true only for one assignment α for $\bar{x}$ and $\bar{x}_\tau$ and thus a final formula with free variable x can be constructed which becomes true for an assignment α', if and only if $\alpha'(x)$ occurs in the set encoded by this assignment α. ◀

From the proof sketch it is easy to see that a dynamic program can also maintain the *size* s of an optimal solution, either implicitly as $|Q|$ for a distinguished relation Q or as $\{v_s\}$. Also, with the adjustments sketched in subsection 6.4, the result can be extended to GSO definable optimisation problems. We can conclude the following corollary from the results of this and the previous section.

▶ **Corollary 10.** *For every k, the Boolean queries* 3-Colourability *and* HamiltonCycle *and the optimisation problems* VertexCover, DominatingSet *and* ShortestPath *are in* DynFO *for graphs of treewidth at most k.*

8 Conclusion

In this paper, we introduced a strategy for maintaining queries by periodically restarting its computation from scratch and limiting the number of change steps that have to be taken into account. This has been captured in the notion of $(\mathcal{C}, f)$-maintainable queries, and we proved that all $(\mathrm{AC}^1, \log n)$-maintainable, almost domain-independent queries are actually in DynFO. As a consequence, decision and optimisation queries definable in MSO- and

[7] We ignore the case that the size could be as large as $|V|$, which can be handled by some additional encoding.

GSO-logic are in DynFO for graphs of bounded treewidth. For this, we stated a Feferman-Vaught-type composition theorem for these logics, which might be interesting in its own right. Though we phrase our results for MSO and GSO for graphs only, their proofs translate swiftly to general relational structures.

We believe that our strategy will find further applications. For instance, it is conceivable that interesting queries on planar graphs, such as the shortest-path query, can be maintained for a bounded number of changes using auxiliary data computed by an AC^1 algorithm (in particular since many important data structures for planar graphs can be constructed in logarithmic space and therefore in AC^1).

References

1. Patricia Bouyer, Vincent Jugé, and Nicolas Markey. Dynamic complexity of parity games with bounded tree-width. *CoRR*, abs/1610.00571, 2016. URL: `http://arxiv.org/abs/1610.00571`.
2. Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in DynFO. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2015. `doi:10.1007/978-3-662-47666-6_13`.
3. Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in DynFO. *CoRR*, abs/1502.07467, 2015. URL: `http://arxiv.org/abs/1502.07467`.
4. Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A strategy for dynamic programs: Start over and muddle through. *CoRR*, abs/1704.07998, 2017. URL: `http://arxiv.org/abs/1704.07998`.
5. Guozhu Dong, Jianwen Su, and Rodney W. Topor. Nonrecursive incremental evaluation of datalog queries. *Ann. Math. Artif. Intell.*, 14(2-4):187–223, 1995. `doi:10.1007/BF01530820`.
6. Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. *ACM Trans. Comput. Log.*, 17(4):25:1–25:18, 2016. `doi:10.1145/2946799`.
7. Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 143–152. IEEE Computer Society, 2010. `doi:10.1109/FOCS.2010.21`.
8. Wouter Gelade, Marcel Marquardt, and Thomas Schwentick. The dynamic complexity of formal languages. *ACM Trans. Comput. Log.*, 13(3):19, 2012. `doi:10.1145/2287718.2287719`.
9. Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.
10. Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.
11. Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997. `doi:10.1006/jcss.1997.1520`.
12. Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. Dynamic complexity under definable changes. In *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy*, pages 19:1–19:18, 2017. `doi:10.4230/LIPIcs.ICDT.2017.19`.
13. Thomas Schwentick and Thomas Zeume. Dynamic complexity: recent updates. *SIGLOG News*, 3(2):30–52, 2016. `doi:10.1145/2948896.2948899`.

14 Saharon Shelah. The monadic theory of order. *Annals of Mathematics*, pages 379–419, 1975. doi:10.2307/1971037.

15 Heribert Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.

16 Volker Weber and Thomas Schwentick. Dynamic complexity theory revisited. *Theory Comput. Syst.*, 40(4):355–377, 2007. doi:10.1007/s00224-006-1312-0.

Definability by Horn Formulas and Linear Time on Cellular Automata*

Nicolas Bacquey[1], Etienne Grandjean[2], and Frédéric Olive[3]

1 **INRIA Lille, Université de Lille, CRIStAL, Villeneuve d'Ascq, France**
`nicolas.bacquey@inria.fr`
2 **Normandie Université, ENSICAEN, CNRS, GREYC, Caen, France**
`etienne.grandjean@unicaen.fr`
3 **Aix Marseille Université, CNRS, LIF UMR 7279, Marseille, France**
`frederic.olive@lif.univ-mrs.fr`

Abstract

We establish an *exact* logical characterization of linear time complexity of cellular automata of dimension d, for any fixed d: a set of pictures of dimension d belongs to this complexity class *iff* it is definable in existential second-order logic restricted to *monotonic* Horn formulas with built-in successor function and $d+1$ first-order variables. This logical characterization is optimal modulo an open problem in parallel complexity. Furthermore, its proof provides a systematic method for transforming an inductive formula defining some problem into a cellular automaton that computes it in linear time.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic, F.4.3 Formal Languages

Keywords and phrases Picture languages, linear time, cellular automata of any dimension, local induction, descriptive complexity, second-order logic, Horn formulas, logic programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.99

1 Introduction

Descriptive complexity provides machine-independent views of complexity classes. Typically, Fagin's Theorem [5] characterizes **NP** as the class of problems definable in existential second-order logic (ESO). Similarly, Immerman-Vardi's Theorem [15] and Grädel's Theorem [8, 9] characterize the class **P** by first-order logic plus least fixed-point, and second-order logic restricted to Horn formulas, respectively. The link between *computational* and *descriptive* complexity can be made as *tight* as possible, i.e. linear time or time $O(n^d)$, for a fixed integer d, can be *exactly* characterized [20, 14, 17, 11]. Two of the present authors have proved in [12, 13] that a problem is recognized in linear time on a non-deterministic cellular automaton of dimension d iff it is definable in ESO logic with built-in successor and $d+1$ first-order variables. Is there such a natural characterization in logic for the more interesting deterministic case? This question motivates the present paper.

A number of algorithmic problems (linear context-free language recognizability, product of integers, product of matrices, sorting...) are computable in linear time on cellular automata of appropriate dimension. For each such problem, the literature describes the algorithm of the cellular automaton in an informal way [2, 16]. In parallel computational models,

* This work was partially supported by ANR AGGREG.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 99; pp. 99:1–99:14

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

algorithms are often difficult to design. However, the problems they solve can often be simply defined inductively. For instance, the product of two integers in binary notation is inductively defined by the classical Horner rule.

The first contribution of this paper is the observation that those inductive processes are naturally formalized by Horn formulas [9]. As our second and main contribution, we notice that for every concrete problem defined by a Horn formula with $d+1$ first-order variables ($d \geq 1$), this inductive computation by Horn rules has a precise *geometrical* characterization: It can be modeled as the displacement of a d-dimensional hyperplane H along some fixed line D in a space of dimension $d+1$. Provided we interpret the line D as a temporal axis, the parallel displacement of H with respect to D coincides with a computation of a d-dimensional cellular automaton. The converse is obvious: a d-dimensional cellular automaton computation can be regarded as the parallel displacement of a d-dimensional hyperplane – its set of cells – along the time axis.

In the next section, a logic is designed which captures these inductive behaviors (see Def. 14). Roughly speaking, it is obtained from the logic ESO-HORN taylored by Grädel to characterize **P**, by restricting both the number of first-order variables and the arity of second-order predicate symbols. Besides, it includes an additional restriction – the 'monotonicity condition' – that reflects the geometrical consideration above-mentioned. We denote this logic by mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$.

Now we can quote the main result of the paper (Thm. 15): a set L of d-pictures can be decided in linear time by a deterministic cellular automaton – written $L \in \mathbf{DLIN}^d_{\text{ca}}$ – if, and only if, it can be expressed in mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$. For short:

$$\mathbf{DLIN}^d_{\text{ca}} = \text{mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1}). \tag{1}$$

A noticeable interest of this result is the constructive method of its proof. In order to design a cellular automaton that computes a problem in linear time, one has to define inductively the problem with a monotonic Horn formula. The normalized form of the formula is automatically obtained: this *is* the program of the cellular automaton[1].

The paper is structured as follows: The next section collects the preliminary definitions and gives a precise statement of our main result. In Sec. 3, we establish the left-to-right inclusion of the identity displayed in (1). The rest of the paper is devoted to the converse inclusion, whose proof is far more involved. In Sec. 4 we build a monotonic Horn formula expressing the language of palindromes (a "toy" example) and deduce from it a cellular automaton that recognizes this language in linear time. Sec. 5 generalizes this construction to any problem defined in mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$, thus completing the proof of (1). In Sec. 6, we conclude by arguing for the optimality of our result.

2 Preliminaries

2.1 Cellular automata, picture languages, linear time

We use the standard terminology of cellular automata as presented in [10].

▶ **Definition 1.** A *cellular automaton of dimension d* (d-CA or CA, for short) is a quadruple $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, where $d \in \mathbb{N}$ is the *dimension* of the automaton, $\mathcal{Q}$ is a finite set whose elements are called *states*, $\mathcal{N}$ is a finite subset of $\mathbb{Z}^d$ called the *neighborhood* of the automaton, and $\delta : \mathcal{Q}^{\mathcal{N}} \to \mathcal{Q}$ is the *local transition function* of the automaton.

[1] For lack of space, this paper gives only one example of this method on a "toy" problem.

A d-CA acts on a grid of dimension d:

▶ **Definition 2.** A *d-dimensional configuration* $\mathcal{C}$ over the set of states $\mathcal{Q}$ is a mapping from $\mathbb{Z}^d$ to $\mathcal{Q}$. The elements of $\mathbb{Z}^d$ will be referred to as *cells* .

▶ **Definition 3.** Given a cellular automaton $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, a configuration $\mathcal{C}$ and a cell $c \in \mathbb{Z}^d$, we call *neighborhood of c in $\mathcal{C}$* the mapping $\mathcal{N}_{\mathcal{C}}(c) : \mathcal{N} \to \mathcal{Q}$ defined by $\mathcal{N}_{\mathcal{C}}(c)(v) = \mathcal{C}(c + v)$.

From the local transition function δ of $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, we can define the *global transition function* of the automaton $\Delta : \mathcal{Q}^{\mathbb{Z}^d} \to \mathcal{Q}^{\mathbb{Z}^d}$ obtained by applying the local rule on each cell, that means $\Delta(\mathcal{C})(c) = \delta(\mathcal{N}_{\mathcal{C}}(c)) = \delta((\mathcal{C}(c + v))_{v \in \mathcal{N}})$, for each cell c.

One identifies the CA $\mathcal{A}$ with its global rule: $\mathcal{A}(\mathcal{C}) = \Delta(\mathcal{C})$ is the image of a configuration $\mathcal{C}$ by the action of $\mathcal{A}$. Moreover, $\mathcal{A}^t(\mathcal{C})$ is the configuration resulting from applying t times the global rule of $\mathcal{A}$ from the initial configuration $\mathcal{C}$.

▶ **Definition 4.** For a given cellular automaton: a state q is *permanent* if a cell in state q remains in this state *regardless* of the states of its neighbors; a state q is *quiescent* if a cell in state q remains in this state if *all* its neighbors are also in state q.

The natural inputs of cellular automata of dimension d are *d-pictures*:

▶ **Definition 5.** Let Σ be a finite alphabet. For integers $d, n \geq 1$, a *picture* of *dimension* d (*d-picture*) and *side* n *over* Σ is a mapping $p : [\![1, n]\!]^d \to \Sigma$. We denote by $\Sigma^{(d)}$ the set of d-pictures over Σ. Any subset of $\Sigma^{(d)}$ is called a *d-picture language* over Σ.

▶ Remark. d-picture languages can capture a wide variety of decision problems if the alphabet Σ is sufficiently expressive. For instance, the product problem of boolean square matrices is a 2-picture problem over the three-part alphabet $\Sigma = \{0, 1\}^3$ that consists of square pictures M such that the projection of M over the last part of the alphabet is equal to the product of its projections over the first two parts.

▶ **Definition 6.** Given a picture $p : [\![1, n]\!]^d \to \Sigma$, we define the *picture configuration* associated with p with *permanent* or *quiescent* state[2] $q_0 \notin \Sigma$ as the function $\mathcal{C}_{p,q_0} : \mathbb{Z}^d \to \Sigma \cup \{q_0\}$ such that $\mathcal{C}_{p,q_0}(\mathbf{x}) = p(\mathbf{x})$ if $\mathbf{x} \in [\![1, n]\!]^d$ and $\mathcal{C}_{p,q_0}(\mathbf{x}) = q_0$ otherwise.

▶ **Definition 7.** Given a d-picture language $L \subseteq \Sigma^d$, we say that a cellular automaton $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$ such that $\Sigma \subseteq \mathcal{Q}$ with permanent states q_a and q_r (accepting and rejecting states) *recognizes* L with permanent state (quiescent state, respectively) $q_0 \in \mathcal{Q} \backslash (\Sigma \cup \{q_a, q_r\})$ *in time* $\tau : \mathbb{N} \to \mathbb{N}$ (for short, $\tau(n)$) if for any picture $p : [\![1, n]\!]^d \to \Sigma$, starting from the configuration $\mathcal{C}_{p,q_0}$ at time 0, the state of cell $\mathbf{n} = (n, \ldots, n)$ of $\mathcal{A}$, called the *reference cell*, is q_a or q_r at time $\tau(n)$ with $\mathcal{A}^{\tau(n)}(\mathcal{C}_{p,q_0})(\mathbf{n}) = q_a$ if $p \in L$ and $\mathcal{A}^{\tau(n)}(\mathcal{C}_{p,q_0})(\mathbf{n}) = q_r$ if $p \notin L$.

▶ **Definition 8.** For $d \geq 1$, we call $\mathbf{DLIN}^d_{ca}$ the class of d-picture problems L for which there exist a d-CA $\mathcal{A}$ with quiescent state q_0 and a function $\tau(n) = O(n)$ such that L can be recognized by $\mathcal{A}$ in time $\tau(n)$. Such a problem is said to be *recognizable in linear time* .

The class $\mathbf{DLIN}^d_{\mathrm{ca}}$ is very robust: it is closed under many changes: neighborhoods, precise time/space bounds, input presentation, etc. The proof of the first part of our main result

[2] The condition that each cell outside the input domain $[\![1, n]\!]^d$ remains in a permanent state (resp. quiescent state) q_0 means that the computation space is exactly the set of input cells (resp. is not bounded).

will use the following restrictive characterization which is a consequence of a general linear acceleration theorem[3] (see *e.g.* [10, 19]).

▶ **Lemma 9** ([10]). $\mathbf{DLIN}^d_{ca}$ *is the class of* d*-picture problems that can be recognized in time* $n-1$ *by a* d*-CA of neighborhood* $\mathcal{N}_2 = \{-2,-1,0,1,2\}^d$ *with* permanent *state* q_0.

2.2 Picture structures and monotonic Horn formulas

The local nature of cellular automata acting on pictures is captured by logical formulas acting on first-order structures, the so-called *picture structures*, that represent these pictures. Before defining picture structures, let us detail their signatures. Given a dimension $d \geq 1$ and k alphabets $\Sigma_1, \ldots, \Sigma_k$, we denote by $\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k)$ the signature below:

$$\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k) = \{(Q^1_s)_{s\in\Sigma_1}, \ldots, (Q^k_s)_{s\in\Sigma_k}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred}\}.$$

Here, each Q^i_s is a d-ary relation symbol, min and max are unary relation symbols, and suc and pred are unary function symbols.

▶ **Definition 10.** Let $p_1, \ldots, p_k$ be pictures of respective alphabets $\Sigma_1, \ldots, \Sigma_k$. We assume that the p_i's have the *same dimension* d and the *same side* n. The *picture structure* of the k-tuple $(p_1, \ldots, p_k)$ is the structure of signature $\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k)$ defined as follows:

$$\mathcal{S}(p_1, \ldots, p_k) = \langle [\![1, n]\!], (Q^1_s)_{s\in\Sigma_1}, \ldots, (Q^k_s)_{s\in\Sigma_k}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred}\rangle.$$

Symbols of $\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k)$ are interpreted on $\mathcal{S}(p_1, \ldots, p_k)$ as follows, where we denote the same way a symbol and its interpretation:

- each Q^i_s is the set of cells of p_i labelled by s. Formally: $Q^i_s = \{a \in [\![1, n]\!]^d : p_i(a) = s\}$;
- min and max are the singleton sets $\{1\}$ and $\{n\}$, respectively;
- suc and pred are the successor and predecessor functions: that is $\mathsf{suc}(n) = n$ and $\mathsf{suc}(a) = a + 1$ for $a \in [\![1, n-1]\!]$; $\mathsf{pred}(1) = 1$ and $\mathsf{pred}(a) = a - 1$ for $a \in [\![2, n]\!]$.

In the following, we will freely use the natural notation $x + i$, for any fixed integer $i \in \mathbb{Z}$. It abbreviates $\mathsf{suc}^i(x)$ if $i > 0$, and $\mathsf{pred}^{-i}(x)$ if $i < 0$. For $i = 0$, it represents x.

We will use the usual definitions and notations in logic (see [4, 18, 9]). All formulas considered hereafter belong to *existential second-order logic*. More precisely, we shall focus on the following logic:

▶ **Definition 11.** $\mathrm{ESO}^d(\forall^{d+1}, \mathrm{arity}^{d+1})$ is the class of formulas of the form $\exists \mathbf{R} \forall \mathbf{x} \psi$, where $\mathbf{R} = (R_1, \ldots, R_r)$ is a tuple of $(d+1)$-ary relation symbols, $\mathbf{x} = (x_0, \ldots, x_d)$ is a $(d+1)$-tuple of first-order variables, and ψ is a quantifier-free first-order formula of signature $\mathsf{sg}(d; \mathbf{\Sigma}) \cup \mathbf{R}$ for some tuple of alphabets $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_k)$.

Such a formula involves two sorts of predicate symbols: those of $\mathsf{sg}(d; \mathbf{\Sigma})$ are called *input predicates* and those of $\mathbf{R}$ are called *guessed predicates* .

It is proved in [13] that the above logic exactly characterizes $\mathbf{NLIN}^d_{\mathrm{ca}}$, the non-deterministic counterpart of $\mathbf{DLIN}^d_{\mathrm{ca}}$. The 'inclusion' $\mathbf{DLIN}^d_{\mathrm{ca}} \subseteq \mathrm{ESO}^d(\forall^{d+1}, arity^{d+1})$ immediately follows, but the converse inclusion is quite unlikely, since it entails $\mathbf{DLIN}^d_{\mathrm{ca}} = \mathbf{NLIN}^d_{\mathrm{ca}}$, which in turn implies $\mathbf{P} = \mathbf{NP}$. Nevertheless, this engages us in looking for a logic characterizing $\mathbf{DLIN}^d_{\mathrm{ca}}$ *inside* the logic $\mathrm{ESO}^d(\forall^{d+1}, arity^{d+1})$. A first restriction of this logic is naturally suggested by the Grädel's characterization of $\mathbf{P}$ already mentioned in the introduction:

[3] For such a result, we need to increase the set of states of the automaton.

▶ **Definition 12.** ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ brings together formulas $\exists\mathbf{R}\forall\mathbf{x}\psi$ among ESO$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ whose quantifier-free part ψ is a conjunction of Horn clauses of the form[4] $\alpha_1 \wedge \ldots \wedge \alpha_m \to \alpha_0$ such that:

- each premise $\alpha_1, \ldots, \alpha_m$ is
  - either a *guessed atom* $R(x_0 + i_0, \ldots, x_d + i_d)$ with $R \in \mathbf{R}$ and $i_0, \ldots, i_d \in \mathbb{Z}$,
  - or an *input literal* $I(t_1 + i_1, \ldots, t_q + i_q)$ or $\neg I(t_1 + i_1, \ldots, t_q + i_q)$, with $I \in \mathsf{sg}(d; \mathbf{\Sigma})$, $t_1, \ldots, t_q \in \mathbf{x}$, and $i_0, \ldots, i_q \in \mathbb{Z}$;
- the conclusion literal α_0 is either a 'constant' – the boolean $\bot$ or an input literal – or a guessed atom[5] of the restricted form $R(x_0, \ldots, x_d)$ with $R \in \mathbf{R}$.

We will see that this new logic still contains $\mathbf{DLIN}^d_{\text{ca}}$ but that here again the converse inclusion is unlikely, as argued in Sec. 6. Whence the necessity of a further restriction of the logic, detailed in Def. 14. For now, let us give some motivation for this restriction.

The first-order part of an ESO-HORN$^d(\forall^{d+1}, arity^{d+1})$-formula can be viewed as a program whose execution, on a given picture structure taken as input, computes the guessed predicates from the input ones. Consider for instance the Horn clause $R(x-2, y-1) \wedge R'(x+1, y-2) \to R(x, y)$ built on guessed predicates R and R'. It establishes a dependence between the values of the guessed predicates (taken as a whole) at place (x, y) and their values at place $(x-2, y-1)$, on one hand, and at place $(x+1, y-2)$, on the other hand. This notion is formalized by the definition below.

▶ **Definition 13.** Let $\Phi = \exists\mathbf{R}\forall x_0, \ldots, x_d\psi$ be in ESO-HORN$^d(\forall^{d+1}, arity^{d+1})$. A *nonzero* tuple $(i_0, \ldots, i_d) \in \mathbb{Z}^{d+1}$ is an *induction vector* of Φ if there exists a Horn clause C in ψ and two guessed predicates R, R' in $\mathbf{R}$ such that C includes $R(x_0, \ldots, x_d)$ as its conclusion and $R'(x_0 + i_0, \ldots, x_d + i_d)$ among its hypotheses. The set of induction vectors of Φ is called its *induction system* .

The logic involved in the characterization of $\mathbf{DLIN}^d_{\text{ca}}$ that constitutes the core of this paper is defined as follows:

▶ **Definition 14.** A formula $\Phi \in$ ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ with induction system $\mathcal{S}$ is *monotonic* and we write $\Phi \in$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ if there exist $a_0, \ldots, a_d \in \mathbb{Z}$ such that each induction vector $(v_0, \ldots, v_d) \in \mathcal{S}$ fulfils $a_0v_0 + \cdots + a_dv_d < 0$. This condition is called the *monotonicity condition* .

In other words, there exists a hyperplane $a_0x_0 + \cdots + a_dx_d = 0$, called a *reference hyperplane* of $\mathcal{S}$, such that each vector $(v_0, \ldots, v_d) \in \mathcal{S}$ belongs to the same strict half-space determined by this hyperplane, that means $a_0v_0 + \cdots + a_dv_d < 0$. One also says that $\mathcal{S} \subset \mathbb{Z}^{d+1}$ *satisfies the monotonicity condition* w.r.t. the reference hyperplane.

We are now in a position to state formally the main result of the paper:

▶ **Theorem 15.** *For* $d \geq 1$, $\mathbf{DLIN}^d_{ca} =$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$.

The two 'inclusions' underlying the above characterization are proved in Sec. 3 and 5.

[4] We will always assume that conjunction has priority over implication.

[5] Alternatively, in Horn formulas, 'guessed' predicates and 'guessed' atoms can be called more intuitively 'computed' predicates and 'computed' atoms.

3 $\mathbf{DLIN}_{\mathsf{ca}} \subseteq$ mon-ESO-HORN

▶ **Proposition 16.** *For $d \geq 1$, $\mathbf{DLIN}^d_{ca} \subseteq$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$.*

Proof. Let $L \subseteq \Sigma^{(d)}$ be a d-picture language in $\mathbf{DLIN}^d_{\text{ca}}$. By Lemma 9, there exists a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}_2, \delta)$ of neighborhood $\mathcal{N}_2 = \{-2,-1,0,1,2\}^d$ that recognizes L in time $\tau(n) = n-1$ with *permanent* state q_0. Let $[\![1,n]\!]$ denote the interval of the n instants of the computation of $\mathcal{A}$ on a d-picture of side n; in particular, the initial and final instants are numbered 1 and n, respectively. We are going to construct a formula $\Phi \in$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ that defines L, i.e., that expresses that the computation of $\mathcal{A}$ on a d-picture p accepts it. It is of the form $\Phi \equiv \exists (R_s)_{s\in\mathcal{Q}} \forall t \forall \mathbf{c} \psi$ where $\mathbf{c}$ denotes the d-tuple of variables $(c_1, \ldots, c_d)$ and, for $s \in \mathcal{Q}$, the intended meaning of the guessed atom $R_s(t, \mathbf{c})$ is the following: at the instant t, the cell $\mathbf{c}$ is in the state s. For simplicity of notation, let us assume $d = 1$. Also assume $n \geq 5$. The quantifier-free part ψ of Φ is the conjunction of three kinds of Horn clauses:

1. the *initialization clauses*: for each $s \in \Sigma$, the clause $\mathsf{min}(t) \wedge Q_s(c) \to R_s(t,c)$;
2. five kinds of *computation clauses* that compute the state at instant $t > 1$ of any cell c according to its possible neighborhoods for $\mathcal{N}_2 = \{-2,-1,0,1,2\}$:
   (i) $c = 1$; (ii) $c = 2$; (iii) general case $c \in [\![3, n-2]\!]$; (iv) $c = n-1$; (v) $c = n$.
   Let us consider the general case: for any 5-tuple of states $(s_{-2}, s_{-1}, s_0, s_1, s_2) \in (Q - \{q_0, q_a, q_r\})^5$, the clause
   $$\left(\begin{array}{c} R_{s_{-2}}(t-1, c-2) \wedge R_{s_{-1}}(t-1, c-1) \wedge \\ R_{s_0}(t-1, c) \wedge R_{s_1}(t-1, c+1) \wedge R_{s_2}(t-1, c+2) \end{array} \right) \to R_{\delta(s_{-2}, s_{-1}, s_0, s_1, s_2)}(t, c)$$
   computes the state at any instant $t > 1$ of any cell c in the interval $[\![3, n-2]\!]$, which can be tested by the use of $\neg\mathsf{min}()$ and $\neg\mathsf{max}()$ in the premises;
3. the *accepting clause* $R_{q_r}(t,c) \to \bot$, which says that the computation does not reject, and hence accepts, since by hypothesis each computation of $\mathcal{A}$ either accepts or rejects.

By construction, Φ belongs to ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ and the induction system is $\{-1\} \times \{-2,-1,0,1,2\}^d$, which has a reference hyperplane of equation $t = 0$. Hence, $\Phi \in$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$, which completes the proof. ◀

4 From the formula to the automaton: the example of palindromes

As the proof of the inclusion mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1}) \subseteq \mathbf{DLIN}^d_{\text{ca}}$ we will give in Sec. 5 is much more elaborate than its converse, we first give in this section its main ideas which are essentially of geometrical nature by now presenting the inductive definition of a "toy" problem by a monotonic Horn formula from which we will derive a cellular automaton that recognizes the problem.

Let PALINDROME(Σ) denote the language of palindromes over a fixed alphabet Σ.

4.1 A monotonic Horn formula defining the language of palindromes

Let us prove that PALINDROME(Σ) is definable in mon-ESO-HORN$^1(\forall^2, \text{arity}^{\,2})$. In addition to the set of input unary predicates $\mathtt{Input} = \{\mathsf{min}, \mathsf{max}, (Q_s)_{s\in\Sigma}\}$ involved in the picture structure that represents a word $w = w_1 w_2 \ldots w_n \in \Sigma^*$, we need to consider three guessed binary predicates symbols $R_=$, $R_<$ and R_{noPal}. The first one is enforced to contain the equality relation. It is used to inductively map the second one on the usual strict linear order over $[\![1,n]\!]$. This is done with the clauses $\theta_1, \ldots, \theta_5$ below:

- $\theta_1 = \mathsf{min}(x) \wedge \mathsf{min}(y) \to R_{=}(x,y)$;
- $\theta_2 = \neg\mathsf{min}(x) \wedge \neg\mathsf{min}(y) \wedge R_{=}(x-1,y-1) \to R_{=}(x,y)$;
- $\theta_3 = \neg\mathsf{max}(x) \wedge R_{=}(x+1,y) \to R_{<}(x,y)$;
- $\theta_4 = \neg\mathsf{max}(x) \wedge R_{<}(x+1,y) \to R_{<}(x,y)$;
- $\theta_5 = R_{<}(x,x) \to \bot$.

The (set of) clauses θ_6 and θ_7 below inductively define R_{noPal} as the set of couples $(x,y) \in [\![1,n]\!]^2$ that fulfill: $x < y$ and the factor $w_x \dots w_y$ of the input word w *is not* a palindrome. Then the clause θ_8 forces w to be a palindrome:

- $\theta_6 = R_{<}(x,y) \wedge Q_s(x) \wedge Q_{s'}(y) \to R_{\mathrm{noPal}}(x,y)$, for all $s \neq s'$ in Σ;
- $\theta_7 = R_{<}(x,y) \wedge R_{\mathrm{noPal}}(x+1,y-1) \to R_{\mathrm{noPal}}(x,y)$;
- $\theta_8 = \mathsf{min}(x) \wedge \mathsf{max}(y) \wedge R_{\mathrm{noPal}}(x,y) \to \bot$.

In conclusion, Palindrome(Σ) is defined by the following formula Φ_{pal}, over the structure $\mathcal{S}(w) = \langle [\![1,n]\!], (Q_s)_{s\in\Sigma}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred}\rangle$ associated with an input word $w = w_1 \dots w_n$:

$$\Phi_{\mathrm{pal}} \equiv \exists R_{=}, R_{<}, R_{\mathrm{noPal}} \forall x, y \bigwedge_{i\leq 8} \theta_i.$$

Moreover, Φ_{pal} belongs to ESO-HORN$^1(\forall^2, \mathrm{arity}\,^2)$ and has $\mathcal{S} = \{(-1,-1),(1,0),(1,-1)\}$ as its induction system (see Def. 13) Clearly, the system $\mathcal{S}$ satisfies the monotonicity condition of Def. 14 with the line of equation $-x+2y=0$ as its reference hyperplane. It follows:

▶ **Proposition 17.** Palindrome(Σ) $\in$ mon-ESO-HORN$^1(\forall^2, \mathrm{arity}\,^2)$.

4.2 From Φ_{pal} to $\mathcal{A}_{\mathsf{pal}}$

It remains to transform the formula Φ_{pal} above into a one-dimensional cellular automaton $\mathcal{A}_{pal}$ that recognizes the language Palindrome(Σ). For sake of simplicity, we first ignore the input literals and only take account of guessed atoms in the Horn clauses θ_i. Notice that in each clause whose conclusion is a guessed atom $R(x,y)$, $R \in \{R_{=}, R_{<}, R_{\mathrm{noPal}}\}$, the guessed atoms occurring as premises have one of the following forms:

$$R'(x,y), \quad R'(x+1,y), \quad R'(x+1,y-1), \quad R'(x-1,y-1).$$

Intuitively, if one regards the set $D_t = \{(x,y) \in [\![1,n]\!]^2 \mid -x+2y = t\}$ as the line of cells of a one-dimensional CA at instant t, then the conjunction of the above clauses θ_i can be regarded as the transition function of such a CA (see Fig. 1). More formally, in order to introduce the *time* parameter t, we eliminate one of the variables, x for example, and we regard the other variable, y, as the *space* variable c. That is, one makes the change of variables[6] : $t = -x+2y$; $c = y$.

Let us now explain how the automaton $\mathcal{A}_{\mathrm{pal}}$ to be constructed can take account of the input literals. For each point $(x,y) \in [\![1,n]\!]^2$, we call state(x,y) the tuple of boolean values of all input and output atoms on x and y. That is,

$$\mathrm{state}(x,y) = \begin{pmatrix} \mathsf{min}(x),\ \mathsf{min}(y),\ \mathsf{max}(x),\ \mathsf{max}(y), \\ (Q_s(x))_{s\in\Sigma},\ (Q_s(y))_{s\in\Sigma}, \\ R_{=}(x,y),\ R_{<}(x,y),\ R_{\mathrm{noPal}}(x,y) \end{pmatrix},$$

where the values $R_{=}(x,y)$, $R_{<}(x,y)$, $R_{\mathrm{noPal}}(x,y)$, are deduced by the Horn formula.

[6] There is an analogy between our method and the so-called *loop-skewing* or *polytope/polyhedron* method in compilation and parallel algorithms [6, 1, 7].

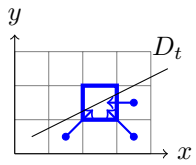


Figure 1 Induction system for guessed atoms before the change of variables.

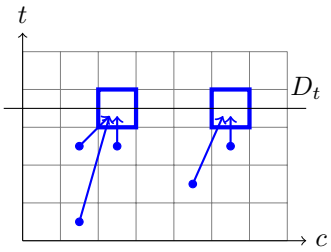


Figure 2 Induction system after the change of variables for guessed atoms (left) and input atoms (right).

We will now discuss the information transit in $\mathcal{A}_{\text{pal}}$; the initialization and termination of the computation will be discussed later. By the change of variables $(x, y) \mapsto (t = -x+2y, c = y)$ whose converse is the function $(t, c) \mapsto (x = -t + 2c, y = c)$, each input atom of the form $I(x)$ becomes $I(-t + 2c)$ and each input atom of the form $I(y)$ becomes $I(c)$. The CA we construct has to memorize in each cell c at instant t the boolean values $I(c)$ and $I(-t + 2c)$, for $I \in \texttt{Input}$. This can be realized as follows:

(a) For each $I(c)$ (former $I(y)$): the CA *conserves* on cell c the boolean value $I(c)$ from an instant $t - 1$ to the next instant t;

(b) For each $I(-t + 2c)$ (former $I(x)$): because of the identity $-t + 2c = -(t - 2) + 2(c - 1)$, whence $I(-t + 2c) \equiv I(-(t - 2) + 2(c - 1))$, the CA only has to *move* to each cell c at instant t the boolean value $I(-(t - 2) + 2(c - 1))$ that is present at instant $t - 2$ on the cell $c - 1$.

All in all, the state of each point $P = (t, c) = (-x + 2y, y)$ is determined by the states of the following points (as shown on Fig. 2):

- $P_1 = (-(x - 1) + 2(y - 1), y - 1) = (t - 1, c - 1)$, $P_2 = (-(x + 1) + 2y, y) = (t - 1, c)$ and $P_3 = (-(x + 1) + 2(y - 1), y - 1) = (t - 3, c - 1)$, because of guessed atoms, and
- $P_2 = (t - 1, c)$ and $P_4 = (t - 2, c - 1)$ because of the above items (a) and (b), respectively, for input atoms.

Hence, the state of a cell c at instant t is determined by the states of: (i) cell $c - 1$ at instant $t - 1$; (ii) cell c at instant $t - 1$; (iii) cell $c - 1$ at instant $t - 3$; (iv) cell $c - 1$ at instant $t - 2$. Figures 1 and 2 below summarize the effects of the change of variables on the induction system.[7]

It seems that we have achieved the design of an automaton of neighborhood $\{-1, 0\}$ that recognizes the language PALINDROME(Σ) in *linear time* since $t = -x + 2y$ and $x, y \in [\![1, n]\!]$ imply $-n + 2 \leq t \leq 2n - 1$. However, it remains to describe both the initialization and the end of the computation.

The result and the initialization of the computation

The result of the computation is `accept` or `reject` according to whether $\mathcal{S}(w)$ does or does not satisfy the formula Φ_{pal}, where $\mathcal{S}(w)$ is the structure $\langle [\![1, n]\!], (Q_s)_{s\in\Sigma}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred}\rangle$

[7] At first glance, Conditions (iii) and (iv) seem to contradict the requirement that the state of any cell c of a CA at instant t should be determined by the *only* states of its neighbour cells at the *previous* instant $t - 1$. However, we can overcome the problem by using the ability of a cell to memorize at any instant t its states at instants $t - 1$ and $t - 2$ with a finite number of states.

associated with the input word $w = w_1 \dots w_n$. As this is testified by the clause $\theta_8 = \mathsf{min}(x) \wedge \mathsf{max}(y) \wedge R_{\mathrm{noPal}}(x, y) \to \bot$, on the point of coordinates $(x = 1, y = n)$ which become after the change of variables $(t = -x + 2y = 2n - 1,\ c = y = n)$, the acceptance/rejection can be read on the state of cell $c = n$ at the instant $t = 2n - 1$ so that the final state q_a or q_r is obtained at the following instant $2n$.

The initialization of the computation requires some care in connection with the items (a) and (b) of the previous paragraph, about the input bits:

(1) *Initializing each* $I(c)$ (former $I(y)$): The state of each cell $c \in [\![1, n]\!]$ at the instant just before $-n + 2$, i.e. at instant $-n + 1$, should store the boolean value $I(c)$, for each $I \in \mathtt{Input}$.

(2) *Initializing each* $I(-t + 2c)$ (former $I(x)$): Because of the correspondence $x = -t + 2c$ or, equivalently, $c = (x + t)/2$, for all $x \in [\![1, n]\!]$, the boolean value $I(x)$ should be stored in the state of the cell $c = (x+t)/2$ at the maximal instant $t < -n+2$ such that $(x+t)/2$ is an *integer*; that is the cell $c = (x - n)/2$ at instant $-n$ if $x - n \equiv 0 \pmod 2$ and $c = (x - n + 1)/2$ at instant $-n + 1$ if $x - n \equiv 1 \pmod 2$: see Fig. 3.

The two configurations at the successive instants $-n$ and $-n + 1$ described in items (1) and (2) are called *initialization* configurations. By construction, the space of both configurations – their informative cells, i.e. those in non-quiescent states – is included in the interval $[\![-\lceil n/2 \rceil + 1, n]\!]$.

According to our conventions, the initial configuration of the automaton should be the configuration $\mathcal{C}_{w,q_0}$ associated with the input word w, as defined in Def. 6. However, one can design a routine which, starting from configuration $\mathcal{C}_{w,q_0}$ (with quiescent state q_0), computes the two initialization configurations by using classical techniques of signals in CA's (such as seen in [3]) as shown on Fig. 4. Once each cell contains the right input information, the proper computation can begin. This start will be triggered by a synchronization of all cells of the interval $[\![-\lceil n/2 \rceil + 1, n + 1]\!]$ at time $-n + 1$. The subject of synchronization on cellular automata has been extensively studied (see [21]), here we merely assert that this process can be performed in parallel with no time increase. By a careful examination of this figure, we precisely observe that this precomputation is performed on the interval of cells $[\![-\lceil n/2 \rceil + 1, n + 1]\!]$ during the time interval $[\![-3n, -n + 1]\!]$.[8]

We have now achieved the design of a cellular automaton $\mathcal{A}_{\mathrm{pal}}$ that recognizes in *linear time* the language $\textsc{Palindrome}(\Sigma)$ from the monotonic Horn formula Φ_{pal} that defines it.

5 mon-ESO-HORN $\subseteq$ $\mathbf{DLIN_{ca}}$

The main problem we have to deal with in the general case as in the previous example is the integration of the input to the computation of the CA to be constructed. For that purpose, we will need the following technical lemma whose proof is easy and left to the reader:

▶ **Lemma 18.** *Let* $\mathcal{S} \subset \mathbb{Z}^{d+1}$ *be an induction system satisfying the monotonicity condition w.r.t. some reference hyperplane. Then,* $\mathcal{S}$ *has another reference hyperplane of equation* $a_0x_0 + \dots + a_dx_d = 0$ *where each coefficient* a_i *(*$i \in [\![0, d]\!]$*) is a non-zero integer.*

[8] Notice that our numbering of instants is not canonical. It is only a convenient time scale for describing our algorithm. In particular, the initial instant of the (pre)-computation of the upper part of Fig. 4 is $-2n$ when n is *even* and $-2n - 1$ when n is *odd*, and the initial instant of the two signals of the lower part of Fig. 4 is $-3n$. We let the reader imagine the variants of Fig. 3 and Fig. 4 for the *odd* case.

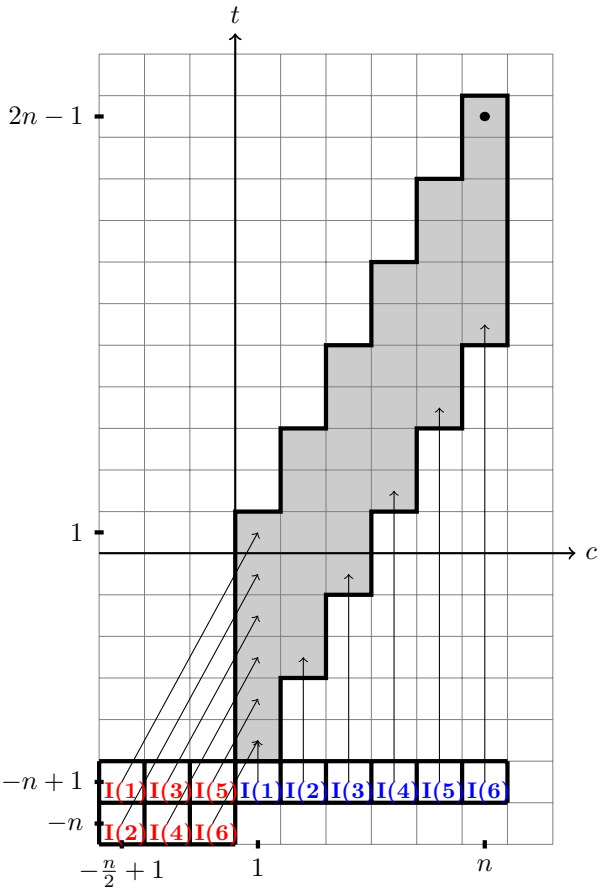


Figure 3 Initial positions and translation vectors for $I(x) = I(-t+2c)$ (in red) and $I(y) = I(c)$ (in blue) when n is *even* (here $n = 6$). The gray parallelogram is where the induction actually happens. The result of the computation lies at the upper right cell.

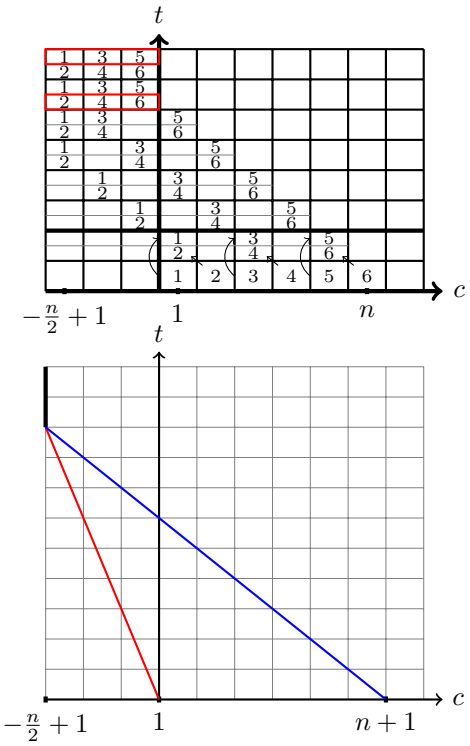


Figure 4 The linear precomputation of $I(x)$ can be done by stacking the information of the cells in the odd columns, then packing it to the left against a "wall" at $c = -\frac{n}{2} + 1$ (n *even*). The bottom figure shows how the wall can be constructed in linear time with two signals of slope $-\frac{1}{3}$ (*resp.* -1) starting in cell 1 (*resp.* $n+1$) at instant $-3n$.

We are now ready to prove the most difficult inclusion of Thm. 15:

▶ **Proposition 19.** *For each* $d \geq 1$, mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{d+1}) \subseteq \mathbf{DLIN}^d_{ca}$.

Proof. Let L be a d-picture language defined by a formula $\Phi \equiv \exists R_1 \dots \exists R_r \forall x_0 \dots \forall x_d\ \psi$ in mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{d+1})$ with an induction system $\mathcal{S}$ and a reference hyperplane $a_0x_0 + a_1x_1 + \dots + a_dx_d = 0$, with coefficients $a_i \in \mathbb{Z}^*$ for all $i \in [\![0, d]\!]$, as justified by Lemma 18. For simplicity of notation, assume all the coefficients a_i are positive.

First, for sake of simplicity, let us ignore the input literals and take account of the only guessed atoms $R_i(x_0 + i_0, \dots, x_d + i_d)$ in the Horn clauses. Intuitively, if one regards the hyperplane $H_t = \{(x_0, x_1, \dots, x_d) \in [\![1, n]\!]^{d+1} \mid a_0x_0 + a_1x_1 + \dots + a_dx_d = t\}$ as the set of cells of a d-dimensional CA at instant t, then the conjunction of Horn clauses ψ can be

regarded as the transition function of such a CA. More formally, in order to introduce the *time* parameter t, we eliminate one of the variables, x_0 for example, and we regard the other variables, $x_1, \dots, x_d$, as the *space* variables, i.e. the respective d coordinates $c_1, \dots, c_d$ of a cell. More precisely, one makes the following change of variables:

$$\left(\begin{array}{l} t = a_0x_0 + \cdots + a_dx_d, \\ c_1 = x_1, \dots, c_d = x_d \end{array} \right), \text{ whose converse is } \left(\begin{array}{l} x_0 = (t - a_1c_1 - \cdots - a_dc_d)/a_0, \\ x_1 = c_1, \dots, x_d = c_d \end{array} \right).$$

As in Section 4.2, we associate with each point $\mathbf{x} = (x_0, \dots, x_d) \in [\![1, n]\!]^{d+1}$, the tuple $\text{state}(\mathbf{x})$ of boolean values of all input and guessed atoms on $\mathbf{x}$. That is,

$$\text{state}(\mathbf{x}) = \left((I(\mathbf{u}))_{\substack{I \in \texttt{Input}, \\ \mathbf{u} \subsetneq \mathbf{x}}}, \ (R(\mathbf{x}))_{R \in \texttt{Guess}} \right).$$

Here, we denote by `Input` (resp. `Guess`) the set of input (resp. guessed) predicates occurring in the formula. Furthermore, $u \in \mathbf{x}$ means that u is any variable among $x_0, \dots, x_d$, while $\mathbf{u} \subsetneq \mathbf{x}$ means that $\mathbf{u}$ is any m-tuple built from those variables, where $m \leq d$ is the arity of I. Besides, the values of the guessed litterals $R(\mathbf{x})$, $R \in \texttt{Guess}$, are deduced by the Horn formula $\forall \mathbf{x} \psi$.

If one ignores the input literals, the state of each point

$$P = (t, c_1, \dots, c_d) = \left(\sum_{j=0}^{d} a_j x_j, \ x_1, \dots, x_d \right)$$

is determined by the states of the points

$$P_v = \left(\sum_{j=0}^{d} a_j (x_j + v_j), \ x_1 + v_1, \dots, x_d + v_d \right) = \left(t + \sum_{j=0}^{d} a_j v_j, \ c_1 + v_1, \dots, c_d + v_d \right)$$

for each vector $v = (v_0, \dots, v_d)$ of the induction system $\mathcal{S}$. In other words the state of the cell $(c_1, \dots, c_d)$ at instant t is determined by the states of the cells $(c_1 + v_1, \dots, c_d + v_d)$ at the respective *previous* instants $t + \sum_{j=0}^{d} a_j v_j$ for the vectors $v = (v_0, \dots, v_d) \in \mathcal{S}$. (Recall that $\sum_{j=0}^{d} a_j v_j < 0$, by hypothesis.)

Let us now explain how the CA we construct can take account of the input atoms, i.e. let us describe how the CA moves the input bits. The crucial point is that at least one of the $d + 1$ variables $x_0, \dots, x_d$ does *not* occur in each input atom because the arity of each input predicate is at most d. This missing variable is used as a 'transport variable' of the values of the concerned input atom. As a *generic* example, let us consider the input atom $I(x_0, x_2, \dots, x_d)$ where I is an input predicate of arity d and where the variable x_1 does not occur[9]. After the above-mentioned change of variables, this atom becomes

$$I(\frac{1}{a_0}(t - a_1c_1 \cdots - a_dc_d), \ c_2, \dots, c_d).$$

Because of the identity $(t - a_1) - a_1(c_1 - 1) - a_2c_2 \cdots - a_dc_d = t - a_1c_1 - a_2c_2 \cdots - a_dc_d$ the automaton only has to *move* to each cell $(c_1, c_2, \dots, c_d)$ at instant t the boolean value

$$I(\frac{1}{a_0}((t - a_1) - a_1(c_1 - 1) - a_2c_2 \cdots - a_dc_d), \ c_2, \dots, c_d)$$

[9] As we have seen in previous examples the case where one variable among $x_2, \dots, x_d$ does not occur in an input atom is similar; the case where x_0 does not occur or the case where the arity of the input predicate is less than d are easier to deal with as we have also seen.

which is stored at instant $t - a_1$ in the state of cell $(c_1 - 1, c_2, \ldots, c_d)$. In terms of cellular automaton, the values of the input atom $I(x_0, x_2, \ldots, x_d)$ are moved/transmitted by linear parallel "signals" which cover all the inductive space $[\![1, n]\!]^{d+1}$.

Time and initialization of the computation: Since the $d+1$ original variables $x_0, \ldots, x_d$ lie in $[\![1, n]\!]$, the domain of the time variable $t = a_0x_0 + \cdots + a_dx_d$ is $[\![A, An]\!]$, where $A = a_0 + \cdots + a_d$. As a consequence, the equation of the cell hyperplane at the *initial* instant (resp. *final* instant) in the space-time diagram is $t = A$ (resp. $t = An$)[10].

The initialization of the input values (input "signals") before the instant $t = A$ is the most delicate/technical part of the computation. It is sufficient to describe the initialization of the values of the input "signals" for our generic example[11] of input atom $\alpha \equiv I(x_0, x_2, \ldots, x_d)$ or, equivalently, $\alpha \equiv I(\frac{1}{a_0}(t - a_1c_1 - \cdots - a_dc_d), c_2, \ldots, c_d)$, for which c_1 (that is the missing variable x_1 of α) is the "transport" variable. To give the reader the geometric intuition of the following construction in the general case we invite her to consult Fig. 3 and 4 of Sec. 4.2 in the particular case of atom $\alpha \equiv I(x) \equiv I(-t + 2c)$ of the formula that defines Palindrome.

Because of the correspondence $t = a_0x_0 + a_1c_1 + a_2x_2 \cdots + a_dx_d$ or, equivalently, $c_1 = (t - a_0x_0 - a_2x_2 \cdots - a_dx_d)/a_1$, with $c_2 = x_2, \ldots, c_d = x_d$, for all $(x_0, x_2, \ldots, x_d) \in [\![1, n]\!]^d$, the boolean value $I(x_0, x_2, \ldots, x_d)$ should be stored – for the initialization of its input "signal" – in the state of the cell $(c_1, x_2, \ldots, x_d)$ such that $c_1 = (t_0 - a_0x_0 - a_2x_2 \cdots - a_dx_d)/a_1$ at the maximal instant $t_0 < A$ (depending on the tuple $(x_0, x_2, \ldots, x_d)$) such that the quotient $(t_0 - a_0x_0 - a_2x_2 \cdots - a_dx_d)/a_1$ is an *integer*. Let i be the integer in $[\![0, a_1 - 1]\!]$ such that $A - a_0x_0 - a_2x_2 \cdots - a_dx_d \equiv -i \pmod{a_1}$. It is easy to verify that $t_0 = A - a_1 + i$. So, the boolean value $I(x_0, x_2, \ldots, x_d)$ should be stored/initialized at the instant $t_0 = A - a_1 + i$ in (the state of) the cell $(c_1, x_2, \ldots, x_d)$ where $c_1 = (A - a_1 + i - a_0x_0 - a_2x_2 \cdots - a_dx_d)/a_1$: see Fig. 3.

Note that for the atom $\alpha \equiv I(x_0, x_2, \ldots, x_d)$, there are a_1 distinct "initialization" configurations in the respective a_1 hyperplanes H_{t_0}, where $t_0 = A - a_1 + i$ with $i \in [\![0, a_1 - 1]\!]$, according to the possible values of the function $f(x_0, x_2, \ldots, x_d) = A - a_0x_0 - a_2x_2 \cdots - a_dx_d$ modulo a_1. Furthermore, one can verify that, by construction, the space of the "initialization" configurations – their informative cells, in non-quiescent states – is included in a hypercube of the form $[\![-bn, bn]\!]^d$, for some constant integer $b > 0$.

Pre-computation and end of computation: The initial configuration of a d-CA that recognizes the d-picture language L should be the *picture configuration* $\mathcal{C}_{p,q_0}$ where p is the input picture. Therefore, there should be a *pre-computation* starting from $\mathcal{C}_{p,q_0}$ that computes the "initialization" configurations of the input atoms of Φ. By the classical technique of signals in CAs (see [3]) we have exemplified above in the case of Palindrome (see Fig. 4), this can be done in linear space and linear time.

Similarly, the result of the computation should be given by the accept/reject state, q_a or q_r, in the *reference* cell $\mathbf{n} = (n, \ldots, n)$. This is realized in linear time by gathering in the reference cell the possible contradictions deduced in cells for Horn clauses.

For lack of space, we have omitted to deal with loops in Horn clauses: the possible presence of guessed atoms of the form $R(t, \mathbf{c})$, i.e. without predecessor/successor functions, *both as conclusions and as hypotheses* of clauses of monotonic Horn formulas seemingly

[10] In the sequel, A always denote the sum $a_0 + \cdots + a_d$. Also, recall that each a_i is positive.
[11] Here again, all the other examples have either exactly the same treatment or a simpler one.

contradicts the "strict monotonicity" of the induction. We leave the reader to cope with this point. This achieves the proof of Prop. 19 and Thm. 15. ◀

6 Optimality of our main result

It is natural to ask whether the monotonicity condition can be removed or weakened in our main result. This is unlikely because it would imply (as the reader can convince herself) the following time-space trade-off which would be a breakthrough in computational complexity:

▶ **Proposition 20.** *If we had* $\mathbf{DLIN}^d_{ca} = \text{ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1})$ *or the weaker equality* $\mathbf{DLIN}^d_{ca} = \text{weak-mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1})$ *for a given* $d > 1$*, then any set of words recognizable by a 1-CA in time* n^d *on* n *cells would be recognizable by a* d*-CA in time* $O(n)$ *on* $O(n^d)$ *cells.*

Here, weak-mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ denotes the variant of the class mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ where the strict inequality $a_0x_0 + \cdots + a_dx_d < 0$ of the monotonicity condition is replaced by the non-strict inequality $a_0x_0 + \cdots + a_dx_d \leq 0$.

References

1. Uday Bondhugula, Albert Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral parallelizer and locality optimizer. In *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 101–113, 2008. doi:10.1145/1375581.1375595.
2. Walter Bucher, II Culik, et al. On real time and linear time cellular automata. *RAIRO, Informatique théorique*, 18(4):307–325, 1984.
3. Marianne Delorme and Jacques Mazoyer. Signals on cellular automata. In Andrew Adamatzky, editor, *Collision-Based Computing*, pages 231–275. Springer London, London, 2002. doi:10.1007/978-1-4471-0129-1_9.
4. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1995.
5. R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings*, pages 43–73, 1974.
6. Paul Feautrier. Some efficient solutions to the affine scheduling problem. part II. multidimensional time. *International Journal of Parallel Programming*, 21(6):389–420, 1992. doi:10.1007/BF01379404.
7. Paul Feautrier and Christian Lengauer. Polyhedron model. In *Encyclopedia of Parallel Computing*, pages 1581–1592. 2011. doi:10.1007/978-0-387-09766-4_502.
8. E. Grädel. Capturing complexity classes by fragments of second order logic. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 – July 3, 1991*, pages 341–352, 1991. doi:10.1109/SCT.1991.160279.
9. E. Grädel, Ph. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. Springer, 2007.
10. A. Grandjean and V. Poupet. A Linear Acceleration Theorem for 2D Cellular Automata on All Complete Neighborhoods. In I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 115:1–115:12, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2016.115.
11. E. Grandjean and F. Olive. Graph properties checkable in linear time in the number of vertices. *J. Comput. Syst. Sci.*, 68(3):546–597, 2004. doi:10.1016/j.jcss.2003.09.002.

12 E. Grandjean and F. Olive. Descriptive complexity for pictures languages. In P. Cégielski and A. Durand, editors, *Computer Science Logic (CSL'12) – 26th International Workshop/21st Annual Conference of the EACSL*, volume 16 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 274–288, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2012.274.

13 E. Grandjean and F. Olive. A logical approach to locality in pictures languages. *J. Comput. Syst. Sci.*, 82(6):959–1006, 2016. doi:10.1016/j.jcss.2016.01.005.

14 E. Grandjean and T. Schwentick. Machine-independent characterizations and complete problems for deterministic linear time. *SIAM J. Comput.*, 32(1):196–230, 2002. doi:10.1137/S0097539799360240.

15 N. Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

16 J. Kari. Basic concepts of cellular automata. In G. Rozenberg, T. Bäck, and J. N. Kok, editors, *Handbook of Natural Computing*, volume 1, pages 3–24. Springer, 2012.

17 C. Lautemann, N. Schweikardt, and T. Schwentick. A logical characterisation of linear time on nondeterministic turing machines. In *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, pages 143–152, 1999. doi:10.1007/3-540-49116-3_13.

18 L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

19 Jacques Mazoyer and Nicolas Reimen. A linear speed-up theorem for cellular automata. *Theoretical Computer Science*, 101(1):59–98, 1992.

20 T. Schwentick. Descriptive complexity, lower bounds and linear time. In *Computer Science Logic, 12th International Workshop, CSL'98, Annual Conference of the EACSL, Brno, Czech Republic, August 24-28, 1998, Proceedings*, pages 9–28, 1998. doi:10.1007/10703163_2.

21 Hiroshi Umeo. Firing squad synchronization problem in cellular automata. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 3537–3574. Springer New York, New York, NY, 2009. doi:10.1007/978-0-387-30440-3_211.

Asynchronous Distributed Automata: A Characterization of the Modal Mu-Fragment*

Fabian Reiter

IRIF, Université Paris Diderot, Paris, France
fabian.reiter@gmail.com

Abstract

We establish the equivalence between a class of asynchronous distributed automata and a small fragment of least fixpoint logic, when restricted to finite directed graphs. More specifically, the logic we consider is (a variant of) the fragment of the modal μ-calculus that allows least fixpoints but forbids greatest fixpoints. The corresponding automaton model uses a network of identical finite-state machines that communicate in an asynchronous manner and whose state diagram must be acyclic except for self-loops. Exploiting the connection with logic, we also prove that the expressive power of those machines is independent of whether or not messages can be lost.

1998 ACM Subject Classification C.2.4 Distributed Systems, F.1.1 Models of Computation, F.4.1 Mathematical Logic

Keywords and phrases Finite automata, distributed computing, modal logic, μ-calculus

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.100

1 Introduction

One of the core disciplines of distributed computing is to design and analyze message passing algorithms that solve graph problems in computer networks. Usually, the problem instance considered in that context is precisely the graph defined by the network in which the computations are performed. All nodes of the network run the same algorithm concurrently, and often make no prior assumptions about the size and topology of the graph. Typical problems that can be solved by such distributed algorithms include graph coloring, leader election, and the construction of spanning trees and maximal independent sets. A comprehensive treatment of the subject can be found in [10] and [11].

The present paper follows up on relatively recent results by Hella et al. and Kuusisto, which establish novel connections between modal logic and some restricted classes of distributed algorithms. These weak types of algorithms, referred to in the following as *distributed automata*, can be represented as deterministic finite-state machines that read sets of states instead of the usual alphabetic symbols. Intuitively, to run a distributed automaton on some node-labeled directed graph G, a separate copy of the same machine is placed on every node and initialized to a state that may depend on the node's label. Each node v communicates with its peers by sending its current state q to every outgoing neighbor, while at the same time collecting the states received from its incoming neighbors into a set S. The successor state of q is then computed as a function of q and S. In particular, this means that v cannot distinguish between two incoming neighbors that share the same state. Acting as a semi-decider, the automaton accepts G at position v precisely if v visits an accepting state at some point in time. Either way, all machines of the network run and communicate forever.

* This work is supported by the DeLTA project (ANR-16-CE40-0007).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 100; pp. 100:1–100:14

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

In [5, 6], Hella et al. have compared several classes of distributed algorithms, of which the weakest uses the restricted communication model described above. Deviating only in nonessential details from their original definition, we can think of those weakest algorithms as *local synchronous* distributed automata. Here, "synchronous" means that all nodes of the network share a global clock, thereby allowing the computation to proceed in an infinite sequence of rounds. In each round, all the nodes compute their next state simultaneously, based on the information collected in the previous round. By the term "local" we mean that the nodes stop changing their state after a constant number of rounds, a usage in accordance with the established terminology of distributed computing (see, e.g., [12]). Equivalently, the state diagram of a local automaton is acyclic as long as we ignore sink states (i.e., states that cannot be left once reached). The work of Hella et al. reveals an intriguing link between distributed computing and modal logic. In particular, it follows immediately from [5, 6, Thm. 1] that the graph properties recognizable by local synchronous automata are precisely those definable in *backward modal logic*, the variant of (basic) modal logic where the usual modal operators are replaced by their backward-looking variants.

Motivated by the preceding result, the connection with modal logic was further investigated by Kuusisto in [7] and [8]. The former paper lifts the constraint of locality imposed in [5, 6], thereby allowing automata with arbitrary state diagrams. These (nonlocal) synchronous automata are then given a logical characterization in terms of a new recursive logic dubbed *modal substitution calculus*. Furthermore, [7, Prp. 7] shows that on finite graphs, synchronous automata can easily recognize all the properties definable in the least fixpoint fragment of the backward μ-calculus. This logic, which we shall refer to simply as the *backward μ-fragment*, extends backward modal logic with a least fixpoint operator that may not be negated. It thus allows to express statements using least fixpoints, but unlike in the full backward μ-calculus, greatest fixpoints are forbidden. On the other hand, the reverse conversion from synchronous automata to the backward μ-fragment is not possible in general. As explained in [7, Prp. 6], it is easy to come up with a synchronous automaton that makes crucial use of the fact that a node can determine whether it receives the same information from all of its incoming neighbors at exactly the same time. Such a behavior cannot be simulated in the backward μ-fragment. By the same token, even the much more expressive monadic second-order logic (MSO) is incomparable with synchronous automata.

Given that the preceding argument relies solely on synchrony, it seems natural to ask whether removing this feature can lead to a distributed automaton model that has the same expressive power as the backward μ-fragment. The present paper provides a positive answer to this question. We introduce several classes of *asynchronous automata* that transfer the standard notion of asynchronous algorithm to the setting of finite-state machines. Basically, this means that we eliminate the global clock from the network, thus making it possible for nodes to operate at different speeds and for messages to be delayed for arbitrary amounts of time, or even be lost. From the syntactic point of view, an asynchronous automaton is the same as a synchronous one, but it has to satisfy an additional semantic condition: its acceptance behavior must be independent of any timing-related issues. Taking a closer look at the automata obtained by translating formulas of the backward μ-fragment, we can easily see that they are in fact asynchronous. Furthermore, their state diagrams are almost acyclic, except that all the states are allowed to have self-loops (not only the sink states). We call this property *quasi-acyclic*. The paper's main contribution is to show that now we can also go in the other direction: every quasi-acyclic asynchronous automaton can be converted into an equivalent formula of the backward μ-fragment. Incidentally, this remains true even if we consider a seemingly more powerful variant of asynchronous automata, where all messages

are guaranteed to be delivered. To illustrate the basic concepts, an example of an automaton and an equivalent formula will be provided in Figure 1, at the end of the next section.

The remainder of this paper is organized as follows: After giving the necessary formal definitions in Section 2, we state and briefly discuss the main result in Section 3. The proof is then developed in the last two sections. Section 4 presents the rather straightforward translation from logic to automata. The reverse translation is given in Section 5, which is a bit more involved and therefore occupies the largest part of the paper.

2 Preliminaries

We denote the set of Boolean values by $\mathbb{2} = \{0, 1\}$, the set of non-negative integers by $\mathbb{N} = \{0, 1, 2, \dots\}$, and the set of positive integers by $\mathbb{N}_{>0} = \mathbb{N} \setminus \{0\}$. With respect to a given set S, we write 2^S for the power set, S^k for the set of k-tuples ($k \in \mathbb{N}$), and $|S|$ for the cardinality. As a special case of k-tuples, $\mathbb{2}^k$ denotes the set of all binary strings of length k. Furthermore, the length of a string x is written as $|x|$.

For $\ell \in \mathbb{N}$, a (finite) *ℓ-bit labeled directed graph*, abbreviated *digraph*, is a structure $G = (V, E, \lambda)$, where V is a finite nonempty set of nodes, $E \subseteq V \times V$ is a set of directed edges, and $\lambda \colon V \to \mathbb{2}^\ell$ is a labeling that assigns a binary string of length ℓ to each node. Isomorphic digraphs are considered to be equal. If v lies in V, we call the pair (G, v) a *pointed digraph*. Moreover, if uv is an edge in E, then u is called an *incoming neighbor* of v.

▶ **Definition 1** (Distributed Automaton). A (distributed) *automaton* with ℓ-bit input is a tuple $A = (Q, \delta_0, \delta, F)$, where Q is a finite set of states, $\delta_0 \colon \mathbb{2}^\ell \to Q$ is an initialization function, $\delta \colon Q \times \mathbb{2}^Q \to Q$ is a transition function, and $F \subseteq Q$ is a set of accepting states.

To run such an automaton A on a digraph G, we regard the edges of G as FIFO buffers. Each buffer vw will always contain a sequence of states previously traversed by node v. An adversary chooses when v evaluates δ to push a new state to the back of the buffer, and when the current first state gets popped from the front. The details are clarified in the following.

A *trace* of an automaton $A = (Q, \delta_0, \delta, F)$ is a finite nonempty sequence $\sigma = q_1 \dots q_n$ of states in Q such that $q_i \neq q_{i+1}$ and $\delta(q_i, S_i) = q_{i+1}$ for some $S_i \subseteq Q$. We say that A is *quasi-acyclic* if its set of traces $\mathfrak{Q}$ is finite. In other words, its state diagram must not contain any directed cycles, except for self-loops.

For any states $p, q \in Q$ and any (possibly empty) sequence σ of states in Q, we define the unary postfix operators first, last, $\mathsf{pushlast}$ and $\mathsf{popfirst}$ as follows: $p\sigma.\mathsf{first} = \sigma p.\mathsf{last} = p$,

$$\sigma p.\mathsf{pushlast}(q) = \begin{cases} \sigma pq & \text{if } p \neq q, \\ \sigma p & \text{if } p = q, \end{cases} \qquad \text{and} \qquad p\sigma.\mathsf{popfirst} = \begin{cases} \sigma & \text{if } \sigma \text{ is nonempty,} \\ p\sigma & \text{if } \sigma \text{ is empty.} \end{cases}$$

An (asynchronous) *timing* of a digraph $G = (V, E, \lambda)$ is an infinite sequence $\tau = (\tau_1, \tau_2, \tau_3, \dots)$ of maps $\tau_t \colon V \cup E \to \mathbb{2}$, indicating which nodes and edges are active at time t, where 1 is assigned infinitely often to every node and every edge. More formally, for all $t \in \mathbb{N}_{>0}$, $v \in V$ and $e \in E$, there exist $i, j > t$ such that $\tau_i(v) = 1$ and $\tau_j(e) = 1$. We refer to this as the *fairness property* of τ. As a restriction, we say that τ is *lossless-asynchronous* if $\tau_t(uv) = 1$ implies $\tau_t(v) = 1$ for all $t \in \mathbb{N}_{>0}$ and $uv \in E$. Furthermore, τ is called the (unique) *synchronous timing* of G if $\tau_t(v) = \tau_t(e) = 1$ for all $t \in \mathbb{N}_{>0}$, $v \in V$ and $e \in E$.

▶ **Definition 2** (Asynchronous Run). Let $A = (Q, \delta_0, \delta, F)$ be a distributed automaton with ℓ-bit input and $\mathfrak{Q}$ be its set of traces. Furthermore, let $G = (V, E, \lambda)$ be an ℓ-bit labeled

digraph and $\tau = (\tau_1, \tau_2, \tau_3, \dots)$ be a timing of G. The (asynchronous) *run* of A on G timed by τ is the infinite sequence $\rho = (\rho_0, \rho_1, \rho_2, \dots)$ of *configurations* $\rho_t \colon V \cup E \to \mathfrak{Q}$, with $\rho_t(V) \subseteq Q$, which are defined inductively as follows, for $t \in \mathbb{N}$, $v \in V$ and $vw \in E$:

$$\rho_0(v) = \rho_0(vw) = \delta_0(\lambda(v)),$$

$$\rho_{t+1}(v) = \begin{cases} \rho_t(v) & \text{if } \tau_{t+1}(v) = 0, \\ \delta\big(\rho_t(v), \{\rho_t(uv).\mathsf{first} \mid uv \in E\}\big) & \text{if } \tau_{t+1}(v) = 1, \end{cases}$$

$$\rho_{t+1}(vw) = \begin{cases} \rho_t(vw).\mathsf{pushlast}(\rho_{t+1}(v)) & \text{if } \tau_{t+1}(vw) = 0, \\ \rho_t(vw).\mathsf{pushlast}(\rho_{t+1}(v)).\mathsf{popfirst} & \text{if } \tau_{t+1}(vw) = 1. \end{cases}$$

If τ is the synchronous timing of G, we refer to ρ as the *synchronous run* of A on G.

Throughout this paper, we assume that our digraphs, automata and logical formulas agree on the number ℓ of labeling bits. An automaton A *accepts* a pointed digraph (G, v) under timing τ if v visits an accepting state at some point in the run ρ of A on G timed by τ, i.e., if there exists $t \in \mathbb{N}$ such that $\rho_t(v) \in F$. If we simply say that A accepts (G, v), without explicitly specifying a timing τ, then we stipulate that ρ is the synchronous run of A on G.

Given a digraph $G = (V, E, \lambda)$ and a class T of timings of G, the automaton A is called *consistent* for G and T if for all $v \in V$, either A accepts (G, v) under every timing in T, or A does not accept (G, v) under any timing in T. We say that A is *asynchronous* if it is consistent for every possible choice of G and T, and *lossless-asynchronous* if it is consistent for every choice where T contains only lossless-asynchronous timings. By contrast, we call an automaton *synchronous* if we wish to emphasize that no such consistency requirements are imposed. Intuitively, all automata can operate in the synchronous setting, but only some of them also work reliably in environments that provide fewer guarantees.

A *digraph property* is a set L of pointed digraphs. We call L the digraph property *recognized* by an automaton A if it consist precisely of those pointed digraphs that are accepted by A. We denote by AA , LA and SA the classes of digraph properties recognizable by asynchronous, lossless-asynchronous and synchronous automata, respectively. Similarly, QAA , QLA and QSA are the corresponding classes recognizable by quasi-acyclic automata.

Turning to logic, let Var be an infinite supply of propositional variables. We define the formulas of *backward modal logic* with ℓ propositional constants by means of the grammar

$$\varphi ::= \bot \mid \top \mid P_i \mid \neg P_i \mid X \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \bar{\Diamond}\varphi \mid \bar{\Box}\varphi\,,$$

where $0 \leq i < \ell$ and $X \in \mathrm{Var}$. Note that this syntax ensures that variables cannot be negated. Given such a formula φ, an ℓ-bit labeled digraph $G = (V, E, \lambda)$ and a variable assignment $\alpha \colon \mathrm{Var} \to 2^V$, we write $[\![\varphi]\!]_{G,\alpha}$ to denote the subset of nodes of G at which φ holds with respect to α. For *atomic propositions* P_i and X, the corresponding semantics are defined by $[\![P_i]\!]_{G,\alpha} = \{v \in V \mid \lambda(v)(i) = 1\}$ and $[\![X]\!]_{G,\alpha} = \alpha(X)$, where $\lambda(v)(i)$ is the i-th bit of $\lambda(v)$. The Boolean constants and connectives are interpreted in the usual way, for instance, $[\![\top]\!]_{G,\alpha} = V$ and $[\![(\varphi \vee \psi)]\!]_{G,\alpha} = [\![\varphi]\!]_{G,\alpha} \cup [\![\psi]\!]_{G,\alpha}$. Finally, the *backward diamond* $\bar{\Diamond}$ and the *backward box* $\bar{\Box}$ represent backward-looking modal operators, with the semantics

$$[\![\bar{\Diamond}\varphi]\!]_{G,\alpha} = \big\{v \in V \;\big|\; u \in [\![\varphi]\!]_{G,\alpha} \text{ for some } u \in V \text{ such that } uv \in E\big\} \quad \text{and}$$
$$[\![\bar{\Box}\varphi]\!]_{G,\alpha} = \big\{v \in V \;\big|\; u \in [\![\varphi]\!]_{G,\alpha} \text{ for all } u \in V \text{ such that } uv \in E\big\}.$$

Traditionally, the modal μ-calculus is defined to comprise individual fixpoints which may be nested. However, it is well-known that we can add simultaneous fixpoints to the

μ-calculus without changing its expressive power, and that nested fixpoints of the same type (i.e., least or greatest) can be rewritten as non-nested simultaneous ones (see, e.g., [3, § 3.7] or [9, § 4.3]). The following definition directly takes advantage of this fact. We shall restrict ourselves to the *μ-fragment of the backward μ-calculus*, abbreviated *backward μ-fragment*, where only least fixpoints are allowed, and where the usual modal operators are replaced by their backward-looking variants. Without loss of generality, we stipulate that each formula of the backward μ-fragment with ℓ propositional constants is of the form

$$\varphi = \mu \begin{pmatrix} X_0 \\ \vdots \\ X_k \end{pmatrix} . \begin{pmatrix} \varphi_0(P_0, \dots, P_{\ell-1}, X_0, \dots, X_k) \\ \vdots \\ \varphi_k(P_0, \dots, P_{\ell-1}, X_0, \dots, X_k) \end{pmatrix},$$

where $X_0, \dots, X_k \in \mathrm{Var}$, and $\varphi_0, \dots, \varphi_k$ are formulas of backward modal logic with ℓ propositional constants that may contain no other variables than $X_0, \dots, X_k$.

For every digraph $G = (V, E, \lambda)$, the tuple $(\varphi_0, \dots, \varphi_k)$ gives rise to an operator $f\colon (2^V)^{k+1} \to (2^V)^{k+1}$ that takes some valuation of $\vec{X} = (X_0, \dots, X_k)$ and reassigns to each X_i the resulting valuation of φ_i. More formally, f maps $\vec{W} = (W_0, \dots, W_k)$ to $(W_0', \dots, W_k')$ such that $W_i' = [\![\varphi_i]\!]_{G,[\vec{X} \mapsto \vec{W}]}$. Here, $[\vec{X} \mapsto \vec{W}]$ can be any variable assignment that interprets each X_i as W_i. A (simultaneous) *fixpoint* of the operator f is a tuple $\vec{W} \in (2^V)^{k+1}$ such that $f(\vec{W}) = \vec{W}$. Since, by definition, variables occur only positively in formulas, the operator f is *monotonic*. This means that $\vec{W} \subseteq \vec{W}'$ implies $f(\vec{W}) \subseteq f(\vec{W}')$ for all $\vec{W}, \vec{W}' \in (2^V)^{k+1}$, where set inclusions are to be understood componentwise (i.e., $W_i \subseteq W_i'$ for each i). Therefore, by virtue of a theorem due to Knaster and Tarski, f has a *least fixpoint*, which is defined as the unique fixpoint $\vec{U} = (U_0, \dots, U_k)$ of f such that $\vec{U} \subseteq \vec{W}$ for every other fixpoint $\vec{W}$ of f. As a matter of fact, the Knaster-Tarski theorem even tells us that $\vec{U}$ is equal to $\bigcap \{\vec{W} \in (2^V)^{k+1} \mid f(\vec{W}) \subseteq \vec{W}\}$, where set operations must also be understood componentwise. Another, perhaps more intuitive, way of characterizing $\vec{U}$ is to consider the inductively constructed sequence of approximants $(\vec{U}^0, \vec{U}^1, \vec{U}^2, \dots)$, where $\vec{U}^0 = (\emptyset, \dots, \emptyset)$ and $\vec{U}^{j+1} = f(\vec{U}^j)$. Since this sequence is monotonically increasing and V is finite, there exists $n \in \mathbb{N}$ such that $\vec{U}^n = \vec{U}^{n+1}$. It is easy to check that $\vec{U}^n$ coincides with the least fixpoint $\vec{U}$. For more details and proofs, see, e.g., [4, § 3.3.1].

Having introduced the necessary background, we can finally establish the semantics of φ with respect to G: the set $[\![\varphi]\!]_G$ of nodes at which φ holds is precisely U_0, the first component of $\vec{U}$. A pointed digraph (G, v) satisfies φ, in symbols $(G, v) \models \varphi$, if $v \in [\![\varphi]\!]_G$. Accordingly, the digraph property *defined* by φ is $\{(G, v) \mid (G, v) \models \varphi\}$, and we denote by $\Sigma_1^{\overline{\mu}}$ the class of all digraph properties defined by some formula of the backward μ-fragment.

As usual, two devices (i.e., automata or formulas) are *equivalent* if they specify (i.e., recognize or define) the same property. Figure 1 provides an example of such an equivalence.

3 Main result

Based on the definitions given in Section 2, asynchronous automata are a special case of lossless-asynchronous automata, which in turn are a special case of synchronous automata.[1] Furthermore, quasi-acyclicity constitutes an additional (possibly orthogonal) restriction on these models. We thus immediately obtain the hierarchy of classes depicted in Figure 2a.

[1] This may seem counterintuitive at first sight, but it is actually consistent with the standard terminology of distributed computing: an asynchronous algorithm can always serve as a synchronous algorithm (i.e., it can be executed in a synchronous environment), but the converse is not true.

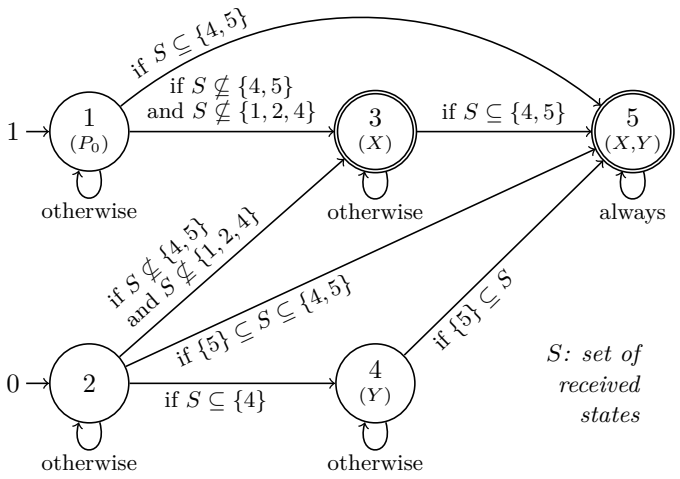
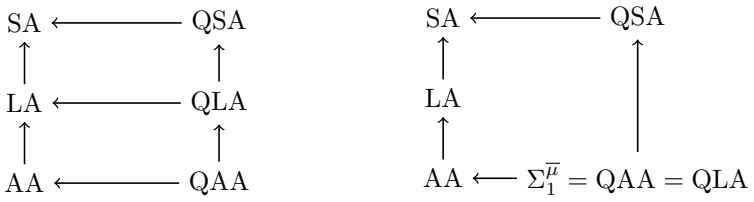


Figure 1 A quasi-acyclic asynchronous distributed automaton equivalent to the formula

$$\mu \begin{pmatrix} X \\ Y \end{pmatrix} . \begin{pmatrix} (P_0 \wedge Y) \vee \overline{\Diamond} X \\ \overline{\Box} Y \end{pmatrix}$$

of the backward μ-fragment. A given pointed 1-bit labeled digraph (G, v) is accepted by this automaton if and only if, starting at v and following G's edges in the backward direction, it is possible to reach some node u labeled with 1 from which it is impossible to reach any directed cycle.



(a) immediate by the definitions **(b)** collapse shown in this paper

Figure 2 Hierarchy of the classes of digraph properties recognizable by distributed automata, depending on whether the automata are synchronous (S), lossless-asynchronous (L), asynchronous (A), or quasi-acyclic (Q). The arrows denote inclusion (e.g., LA ⊆ SA).

Our main result provides a simplification of this hierarchy: the classes QAA and QLA are actually equal to the class of digraph properties definable in the backward μ-fragment. This yields the revised diagram shown in Figure 2b.

▶ **Theorem 3** ($\Sigma_1^{\overline{\mu}}$ = QAA = QLA)**.** *When restricted to finite digraphs, the backward μ-fragment is effectively equivalent to the classes of quasi-acyclic asynchronous automata and quasi-acyclic lossless-asynchronous automata.*

Proof. The forward direction is given by Proposition 4 (in Section 4), which asserts that $\Sigma_1^{\overline{\mu}} \subseteq$ QAA, and the trivial observation that QAA ⊆ QLA. For the backward direction, we use Proposition 7 (in Section 5), which asserts that QLA ⊆ $\Sigma_1^{\overline{\mu}}$. ◀

As stated before, synchronous automata are more powerful than the backward μ-fragment (and incomparable with monadic second-order logic). This holds even if we consider only quasi-acyclic automata, i.e., the inclusion $\Sigma_1^{\overline{\mu}} \subset$ QSA is known to be strict (see [7, Prp. 6]). Moreover, an upcoming paper will show that the inclusion QSA ⊂ SA is also strict.

In contrast, it remains open whether quasi-acyclicity is in fact necessary for characterizing $\Sigma_1^{\overline{\mu}}$. On the one hand, this notion is crucial for our proof (see Proposition 7), but on the other hand, no digraph property separating AA or LA from $\Sigma_1^{\overline{\mu}}$ has been found so far.

4 Computing least fixpoints using asynchronous automata

In this section, we prove the easy direction of the main result. Given a formula φ of the backward μ-fragment, it is straightforward to construct a (synchronous) distributed automaton A that computes on any digraph the least fixpoint $\vec{U}$ of the operator associated with φ. As long as it operates in the synchronous setting, A simply follows the sequence of approximants $(\vec{U}^0, \vec{U}^1, \dots)$ described in Section 2. It is important to stress that the very same observation has previously been made in [7, Prp. 7] (formulated from a different point of view). In the following proposition, we refine this observation by giving a more precise characterization of the obtained automaton: it is always quasi-acyclic and capable of operating in a (possibly lossy) asynchronous environment.

▶ **Proposition 4** ($\Sigma_1^{\overline{\mu}} \subseteq$ QAA). *For every formula of the backward μ-fragment, we can effectively construct an equivalent quasi-acyclic asynchronous automaton.*

Proof. Let $\varphi = \mu(X_0, \dots, X_k).(\varphi_0, \dots, \varphi_k)$ be a formula of the backward μ-fragment with ℓ propositional constants. Without loss of generality, we may assume that the subformulas $\varphi_0, \dots, \varphi_k$ do not contain any nested modal operators. To see this, suppose that $\varphi_i = \overline{\Diamond}\psi$. Then φ is equivalent to $\varphi' = \mu(X_0, \dots, X_i, \dots, X_k, Y).(\varphi_0, \dots, \varphi_i', \dots, \varphi_k, \psi)$, where Y is a fresh propositional variable and $\varphi_i' = \overline{\Diamond}Y$. The operator $\overline{\Box}$ and Boolean combinations of $\overline{\Diamond}$ and $\overline{\Box}$ are handled analogously.

We now convert φ into an equivalent automaton $A = (Q, \delta_0, \delta, F)$ with state set $Q = 2^{\{P_0, \dots, P_{\ell-1}, X_0, \dots, X_k\}}$. The idea is that each node v of the input digraph has to remember which of the atomic propositions $P_0, \dots, P_{\ell-1}, X_0, \dots, X_k$ have, so far, been verified to hold at v. Therefore, we define the initialization function such that $\delta_0(x) = \{P_i \mid x(i) = 1\}$ for all $x \in 2^\ell$. Let us write $(q, S) \models \varphi_i$ to indicate that a pair $(q, S) \in Q \times 2^Q$ satisfies a subformula φ_i of φ. This is the case precisely when φ_i holds at any node v that satisfies exactly the atomic propositions in q and whose incoming neighbors satisfy exactly the propositions specified by S. Note that this satisfaction relation is well-defined in our context because the nesting depth of modal operators in φ_i is at most 1. With that, the transition function of A can be succinctly described by $\delta(q, S) = q \cup \{X_i \mid (q, S) \models \varphi_i\}$. Since $q \subseteq \delta(q, S)$, we are guaranteed that the automaton is quasi-acyclic. Finally, the accepting set is given by $F = \{q \mid X_0 \in q\}$.

It remains to prove that A is asynchronous and equivalent to φ. Let $G = (V, E, \lambda)$ be an ℓ-bit labeled digraph and $\vec{U} = (U_0, \dots, U_k) \in (2^V)^{k+1}$ be the least fixpoint of the operator f associated with $(\varphi_0, \dots, \varphi_k)$. Due to the asynchrony condition, we must consider an arbitrary timing $\tau = (\tau_1, \tau_2, \dots)$ of G. The corresponding run $\rho = (\rho_0, \rho_1, \dots)$ of A on G timed by τ engenders an infinite sequence $(\vec{W}^0, \vec{W}^1, \dots)$, where each tuple $\vec{W}^t = (W_0^t, \dots, W_k^t) \in (2^V)^{k+1}$ specifies the valuation of every variable X_i at time t, i.e., $W_i^t = \{v \in V \mid X_i \in \rho_t(v)\}$. Since A is quasi-acyclic and V is finite, this sequence must eventually stabilize at some value $\vec{W}^\infty$, and each node accepts if and only if it belongs to W_0^∞. Reformulated this way, our task is to demonstrate that $\vec{W}^\infty$ equals $\vec{U}$, regardless of the timing τ.

"$\vec{W}^\infty \subseteq \vec{U}$": We show by induction that $\vec{W}^t \subseteq \vec{U}$ for all $t \in \mathbb{N}$. This obviously holds for $t = 0$, since $\vec{W}^0 = (\emptyset, \dots, \emptyset)$. Now, consider any node $v \in V$ at an arbitrary time t. Let q be the current state of v and S be the set of current states of its incoming neighbors. Depending

on τ, it might be the case that v actually receives some outdated information S' instead of S. However, given that the neighbors' previous states cannot contain more variables than their current ones (by construction), and that variables can only occur positively in each φ_i, we know that $(q, S') \models \varphi_i$ implies $(q, S) \models \varphi_i$. Hence, if v performs a local transition at time t, then the only new variables that can be added to its state must lie in $\{X_i \mid (q, S) \models \varphi_i\}$. On a global scale, this means that $\vec{W}^{t+1} \setminus \vec{W}^t \subseteq f(\vec{W}^t)$. Furthermore, by the induction hypothesis, the monotonicity of f, and the fact that $\vec{U}$ is a fixpoint, we have $f(\vec{W}^t) \subseteq f(\vec{U}) = \vec{U}$. Putting both together, and again relying on the induction hypothesis, we obtain $\vec{W}^{t+1} \subseteq \vec{U}$.

"$\vec{W}^\infty \supseteq \vec{U}$": For the converse direction, we make use of the Knaster-Tarski theorem, which gives us the equality $\vec{U} = \bigcap\{\vec{W} \in (2^V)^{k+1} \mid f(\vec{W}) \subseteq \vec{W}\}$. With this, it suffices to show that $f(\vec{W}^\infty) \subseteq \vec{W}^\infty$. Consider some time $t \in \mathbb{N}$ such that $\vec{W}^{t'} = \vec{W}^\infty$ for all $t' \geq t$. Although we know that every node has reached its final state at time t, the FIFO buffers of some edges might still contain obsolete states from previous times. However, the fairness property of τ guarantees that our customized popfirst operation is executed infinitely often at every edge, while the pushlast operation has no effect because all the states remain unchanged. Therefore, there must be a time $t' \geq t$ from which on each buffer contains only the current state of its incoming node, i.e., $\rho_{t''}(uv) = \rho_{t''}(u)$ for all $t'' \geq t'$ and $uv \in E$. Moreover, the fairness property of τ also ensures that every node v reevaluates the local transition function δ infinitely often, based on its own current state q and the set S of states in the buffers associated with its incoming neighbors. As this has no influence on v's state, we can deduce that $\{X_i \mid (q, S) \models \varphi_i\} \subseteq q$. Consequently, we have $f(\vec{W}^{t'}) \subseteq \vec{W}^{t'}$, which is equivalent to $f(\vec{W}^\infty) \subseteq \vec{W}^\infty$. ◀

5 Capturing asynchronous runs using least fixpoints

This section is dedicated to proving the converse direction of the main result, which will allow us to translate any quasi-acyclic lossless-asynchronous automaton into an equivalent formula of the backward μ-fragment (see Proposition 7). Our proof builds on two concepts: the invariance of distributed automata under backward bisimulation (stated in Proposition 5) and an ad-hoc relation "$\triangleright$" that captures the possible behaviors of a fixed lossless-asynchronous automaton A (in a specific sense described in Lemma 6).

We start with the notion of backward bisimulation, which is defined like the standard notion of bisimulation (see, e.g., [1, Def. 2.16] or [2, Def. 5]), except that edges are followed in the backward direction. Formally, a *backward bisimulation* between two ℓ-bit labeled digraphs $G = (V, E, \lambda)$ and $G' = (V', E', \lambda')$ is a binary relation $R \subseteq V \times V'$ that fulfills the following conditions for all $vv' \in R$:

1. $\lambda(v) = \lambda'(v')$,
2. if $uv \in E$, then there exists $u' \in V'$ such that $u'v' \in E'$ and $uu' \in R$, and, conversely,
3. if $u'v' \in E'$, then there exists $u \in V$ such that $uv \in E$ and $uu' \in R$.

We say that the pointed digraphs (G, v) and (G', v') are *backward bisimilar* if there exists such a backward bisimulation R relating v and v'. It is easy to see that distributed automata cannot distinguish between backward bisimilar structures:

▶ **Proposition 5.** *Distributed automata are invariant under backward bisimulation. That is, for every automaton A, if two pointed digraphs (G, v) and (G', v') are backward bisimilar, then A accepts (G, v) if and only if it accepts (G', v').*

Proof. Let R be a backward bisimulation between G and G' such that $vv' \in R$. Since acceptance is defined with respect to the synchronous behavior of the automaton, we need

only consider the synchronous runs $\rho = (\rho_0, \rho_1, \dots)$ and $\rho' = (\rho'_0, \rho'_1, \dots)$ of A on G and G', respectively. Now, given that the FIFO buffers on the edges of the digraphs merely contain the current state of their incoming node, it is straightforward to prove by induction on t that every pair of nodes $uu' \in R$ satisfies $\rho_t(u) = \rho'_t(u')$ for all $t \in \mathbb{N}$. ◀

We now turn to the mentioned relation "$\triangleright$", which is defined with respect to a fixed automaton. For the remainder of this section, let A denote an automaton (Q, δ_0, δ, F), and let $\mathfrak{Q}$ denote its set of traces. The relation $\triangleright \subseteq (2^{\mathfrak{Q}} \times \mathfrak{Q})$ specifies whether, in a lossless-asynchronous environment, a given trace σ can be traversed by a node whose incoming neighbors traverse the traces of a given set $\mathfrak{S}$. Loosely speaking, the intended meaning of $\mathfrak{S} \triangleright \sigma$ ("$\mathfrak{S}$ enables σ") is the following: Take an appropriately chosen digraph under some lossless-asynchronous timing τ, and observe the corresponding run of A up to a specific time t; if node v was initially in state $\sigma.\mathsf{first}$ and at time t it has *seen* its incoming neighbors traversing precisely the traces in $\mathfrak{S}$, then it is possible for τ to be such that at time t, node v has traversed exactly the trace σ. This relation can be defined inductively: As the base case, we specify that for every $q \in Q$ and $S \subseteq Q$, we have $S \triangleright q.\mathsf{pushlast}(\delta(q, S))$. For the inductive clause, consider a trace $\sigma \in \mathfrak{Q}$ and two finite (possibly equal) sets of traces $\mathfrak{S}, \mathfrak{S}' \subseteq \mathfrak{Q}$ such that the traces in $\mathfrak{S}'$ can be obtained by appending at most one state to the traces in $\mathfrak{S}$. More precisely, if $\pi \in \mathfrak{S}$, then $\pi.\mathsf{pushlast}(p) \in \mathfrak{S}'$ for some $p \in Q$, and conversely, if $\pi' \in \mathfrak{S}'$, then $\pi' = \pi.\mathsf{pushlast}(\pi'.\mathsf{last})$ for some $\pi \in \mathfrak{S}$. We shall denote this auxiliary relation by $\mathfrak{S} \rightrightarrows \mathfrak{S}'$. If it holds, then $\mathfrak{S} \triangleright \sigma$ implies $\mathfrak{S}' \triangleright \sigma.\mathsf{pushlast}(q)$, where $q = \delta(\sigma.\mathsf{last}, \{\pi'.\mathsf{last} \mid \pi' \in \mathfrak{S}'\})$.

The next step is to show (in Lemma 6) that our definition of "$\triangleright$" does indeed capture the intuition given above. To formalize this, we first introduce two further pieces of terminology.

First, the notions of configuration and run can be enriched to facilitate discussions about the past. Let $\rho = (\rho_0, \rho_1, \dots)$ be a run of A on a digraph $G = (V, E, \lambda)$ (timed by some timing τ). The corresponding *enriched run* is the sequence $\hat{\rho} = (\hat{\rho}_0, \hat{\rho}_1, \dots)$ of *enriched configurations* that we obtain from ρ by requiring each node to remember the entire trace it has traversed so far. Formally, for $t \in \mathbb{N}$, $v \in V$ and $e \in E$,

$$\hat{\rho}_0(v) = \rho_0(v), \qquad \hat{\rho}_{t+1}(v) = \hat{\rho}_t(v).\mathsf{pushlast}(\rho_{t+1}(v)) \qquad \text{and} \qquad \hat{\rho}_t(e) = \rho_t(e).$$

Second, we will need to consider finite segments of timings and enriched runs. A *lossless-asynchronous timing segment* of a digraph G is a finite sequence $\tau = (\tau_1, \dots, \tau_r)$ that could be extended to a whole lossless-asynchronous timing $(\tau_1, \dots, \tau_r, \tau_{r+1}, \dots)$. Likewise, for an initial enriched configuration $\hat{\rho}_0$ of G, the corresponding *enriched run segment* timed by τ is the sequence $(\hat{\rho}_0, \dots, \hat{\rho}_r)$, where each $\hat{\rho}_{t+1}$ is computed from $\hat{\rho}_t$ and τ_{t+1} in the same way as for an entire enriched run.

Equipped with the necessary terminology, we can now state and prove a (slightly technical) lemma that will allow us to derive benefit from the relation "$\triangleright$". This lemma essentially states that if $\mathfrak{S} \triangleright \sigma$ holds and we are given enough nodes that traverse the traces in $\mathfrak{S}$, then we can take those nodes as the incoming neighbors of a new node v and delay the messages received by v in such a way that v traverses σ, without losing any messages.

▶ **Lemma 6.** *For every trace $\sigma \in \mathfrak{Q}$ and every finite (possibly empty) set of traces $\mathfrak{S} = \{\pi_1, \dots, \pi_\ell\} \subseteq \mathfrak{Q}$ that satisfy the relation $\mathfrak{S} \triangleright \sigma$, there exist lower bounds $m_1, \dots, m_\ell \in \mathbb{N}_{>0}$ such that the following statement holds true:*

For any $n_1, \dots, n_\ell \in \mathbb{N}_{>0}$ satisfying $n_i \geq m_i$, let G be a digraph consisting of the nodes $(u_i^j)_{i,j}$ and v, and the edges $(u_i^j v)_{i,j}$, with index ranges $1 \leq i \leq \ell$ and $1 \leq j \leq n_i$. If we start from the enriched configuration $\hat{\rho}_0$ of G, where

$$\hat{\rho}_0(u_i^j) = \pi_i, \qquad \hat{\rho}_0(u_i^j v) = \pi_i \qquad \text{and} \qquad \hat{\rho}_0(v) = \sigma.\mathsf{first},$$

then we can construct a (nonempty) lossless-asynchronous timing segment $\tau = (\tau_1, \dots, \tau_r)$ *of* G*, where* $\tau_t(u_i^j) = 0$ *and* $\tau_t(v) = 1$ *for* $1 \le t \le r$*, such that the corresponding enriched run segment* $\hat{\rho} = (\hat{\rho}_0, \dots, \hat{\rho}_r)$ *timed by* τ *satisfies*

$$\hat{\rho}_{r-1}(u_i^j v) = \pi_i.\mathsf{last} \qquad \textit{and} \qquad \hat{\rho}_r(v) = \sigma.$$

Proof. We proceed by induction on the definition of "$\triangleright$". In the base case, where $\mathfrak{S} = \{p_1, \dots, p_\ell\} \subseteq Q$ and $\sigma = q.\mathsf{pushlast}(\delta(q, \mathfrak{S}))$ for some $q \in Q$, the statement holds with $m_1 = \cdots = m_\ell = 1$. This is witnessed by a timing segment $\tau = (\tau_1)$, where $\tau_1(u_i^j) = 0$, $\tau_1(v) = 1$, and $\tau_1(u_i^j v)$ can be chosen as desired.

For the inductive step, we assume that the statement holds for σ and $\mathfrak{S} = \{\pi_1, \dots, \pi_\ell\}$ with some values $m_1, \dots, m_\ell$. Now consider any other set of traces $\mathfrak{S}' = \{\pi'_1, \dots, \pi'_{\ell'}\}$ such that $\mathfrak{S} \rightrightarrows \mathfrak{S}'$, and let $\sigma' = \sigma.\mathsf{pushlast}(q)$, where $q = \delta(\sigma.\mathsf{last}, \{\pi'_k.\mathsf{last} \mid \pi'_k \in \mathfrak{S}'\})$. Since $\mathfrak{S} \triangleright \sigma$, we have $\mathfrak{S}' \triangleright \sigma'$. The remainder of the proof consists in showing that the statement also holds for σ' and $\mathfrak{S}'$ with some large enough integers $m'_1, \dots, m'_{\ell'}$. Let us fix $m'_k = \sum \{m_i \mid \pi_i.\mathsf{pushlast}(\pi'_k.\mathsf{last}) = \pi'_k\}$. (As there is no need to find minimal values, we opt for easy expressibility.)

Given any numbers $n'_1, \dots, n'_{\ell'}$ with $n'_k \ge m'_k$, we choose suitable values $n_1, \dots, n_\ell$ with $n_i \ge m_i$, and consider the corresponding digraph G described in the lemma. Because we have $\mathfrak{S} \rightrightarrows \mathfrak{S}'$, we can assign to each node u_i^j a state p_i^j such that $\pi_i.\mathsf{pushlast}(p_i^j) \in \mathfrak{S}'$. Moreover, provided our choice of $n_1, \dots, n_\ell$ was adequate, we can also ensure that for each $\pi'_k \in \mathfrak{S}'$, there are exactly n'_k nodes u_i^j such that $\pi_i.\mathsf{pushlast}(p_i^j) = \pi'_k$. (Note that nodes with distinct traces $\pi_i, \pi_{i'} \in \mathfrak{S}$ might be mapped to the same trace $\pi'_k \in \mathfrak{S}'$, in case $\pi_{i'} = \pi_i p_i^j$.) It is straightforward to verify that such a choice of numbers and such an assignment of states are always possible, given the lower bounds $m'_1, \dots, m'_{\ell'}$ specified above.

Let us now consider the lossless-asynchronous timing segment $\tau = (\tau_1, \dots, \tau_r)$ and the corresponding enriched run segment $\hat{\rho} = (\hat{\rho}_0, \dots, \hat{\rho}_r)$ provided by the induction hypothesis. Since the $\mathsf{popfirst}$ operation has no effect on a trace of length 1, we may assume without loss of generality that $\tau_t(u_i^j v) = 0$ if $\hat{\rho}_{t-1}(u_i^j v)$ has length 1, for $t < r$. Consequently, if we start from the alternative enriched configuration $\hat{\rho}'_0$, where

$$\hat{\rho}'_0(u_i^j) = \pi_i.\mathsf{pushlast}(p_i^j), \qquad \hat{\rho}'_0(u_i^j v) = \pi_i.\mathsf{pushlast}(p_i^j) \qquad \text{and} \qquad \hat{\rho}'_0(v) = \sigma.\mathsf{first},$$

then the corresponding enriched run segment $(\hat{\rho}'_0, \dots, \hat{\rho}'_r)$ timed by τ can be derived from $\hat{\rho}$ by simply applying "$\mathsf{pushlast}(p_i^j)$" to $\hat{\rho}_t(u_i^j)$ and $\hat{\rho}_t(u_i^j v)$, for $t < r$. We thus get

$$\hat{\rho}'_{r-1}(u_i^j v) = \pi_i.\mathsf{last}.\mathsf{pushlast}(p_i^j) \qquad \text{and} \qquad \hat{\rho}'_r(v) = \sigma.$$

We may also assume without loss of generality that $\tau_r(u_i^j v) = 1$ if $\hat{\rho}'_{r-1}(u_i^j v)$ has length 2, since this does not affect $\hat{\rho}$ and lossless-asynchrony is ensured by $\tau_r(v) = 1$. Hence, it suffices to extend τ by an additional map τ_{r+1}, where $\tau_{r+1}(u_i^j) = 0$, $\tau_{r+1}(v) = 1$, and $\tau_{r+1}(u_i^j v)$ can be chosen as desired. The resulting enriched run segment $(\hat{\rho}'_0, \dots, \hat{\rho}'_{r+1})$ satisfies

$$\hat{\rho}'_r(u_i^j v) = p_i^j = \pi'_k.\mathsf{last} \ \text{ (for some } \pi'_k \in \mathfrak{S}') \qquad \text{and} \qquad \hat{\rho}'_{r+1}(v) = \sigma.\mathsf{pushlast}(q) = \sigma'. \quad \blacktriangleleft$$

Finally, we can put the pieces together and prove the converse direction of Theorem 3:

▶ **Proposition 7** ($\mathrm{QLA} \subseteq \Sigma_1^{\overline{\mu}}$)**.** *For every quasi-acyclic lossless-asynchronous automaton, we can effectively construct an equivalent formula of the backward* μ*-fragment.*

Proof. Assume that $A = (Q, \delta_0, \delta, F)$ is a quasi-acyclic lossless-asynchronous automaton with ℓ-bit input. Since it is quasi-acyclic, its set of traces $\mathfrak{Q}$ is finite, and thus we can afford

to introduce a separate propositional variable X_σ for each trace $\sigma \in \mathfrak{Q}$. Making use of the relation "$\triangleright$", we convert A into an equivalent formula $\varphi = \mu\big[X_0, (X_\sigma)_{\sigma\in\mathfrak{Q}}\big].\big[\varphi_0, (\varphi_\sigma)_{\sigma\in\mathfrak{Q}}\big]$ of the backward μ-fragment, where

$$\varphi_0 = \bigvee_{\substack{\sigma\in\mathfrak{Q}\\ \sigma.\mathsf{last}\in F}} X_\sigma, \tag{a}$$

$$\varphi_q = \bigvee_{\substack{x\in\mathfrak{D}^\ell\\ \delta_0(x)=q}} \Big(\bigwedge_{x(i)=1} P_i \wedge \bigwedge_{x(i)=0} \neg P_i \Big) \qquad \text{for each } q \in Q, \quad \text{and} \tag{b}$$

$$\varphi_\sigma = X_{\sigma.\mathsf{first}} \wedge \bigvee_{\substack{\mathfrak{S}\subseteq\mathfrak{Q}\\ \mathfrak{S}\triangleright\sigma}} \Big(\big(\bigwedge_{\pi\in\mathfrak{S}} \bar{\Diamond} X_\pi\big) \wedge \big(\bar{\Box} \bigvee_{\pi\in\mathfrak{S}} X_\pi\big) \Big) \qquad \text{for each } \sigma \in \mathfrak{Q} \text{ with } |\sigma| \geq 2. \tag{c}$$

Note that this formula can be constructed effectively because an inductive computation of "$\triangleright$" must terminate after at most $|\mathfrak{Q}| \cdot 2^{|\mathfrak{Q}|}$ iterations.

To prove that φ is indeed equivalent to A, let us consider an arbitrary ℓ-bit labeled digraph $G = (V, E, \lambda)$ and the corresponding least fixpoint $\vec{U} = (U_0, (U_\sigma)_{\sigma\in\mathfrak{Q}}) \in (\mathfrak{D}^V)^{|\mathfrak{Q}|+1}$ of the operator f associated with $(\varphi_0, (\varphi_\sigma)_{\sigma\in\mathfrak{Q}})$.

The easy direction is to show that for all nodes $v \in V$, if A accepts (G, v), then (G, v) satisfies φ. For that, it suffices to consider the synchronous enriched run $\hat{\rho} = (\hat{\rho}_0, \hat{\rho}_1, \dots)$ of A on G. (Any other run timed by a lossless-asynchronous timing would exhibit the same acceptance behavior.) As in the proof of Proposition 5, we can simply ignore the FIFO buffers on the edges of G because $\hat{\rho}_t(uv) = \hat{\rho}_t(u).\mathsf{last}$. Using this, a straightforward induction on t shows that every node $v \in V$ satisfies $\{\hat{\rho}_t(u) \mid uv \in E\} \triangleright \hat{\rho}_{t+1}(v)$ for all $t \in \mathbb{N}$. (For $t = 0$, the claim follows from the base case of the definition of "$\triangleright$""; for the step from t to $t+1$, we can immediately apply the inductive clause of the definition.) This in turn allows us to prove that each node v is contained in all the components of $\vec{U}$ that correspond to a trace traversed by v in $\hat{\rho}$, i.e., $v \in U_{\hat{\rho}_t(v)}$ for all $t \in \mathbb{N}$. Naturally, we proceed again by induction: For $t = 0$, we have $\hat{\rho}_0(v) = \delta_0(\lambda(v)) \in Q$, hence the subformula $\varphi_{\hat{\rho}_0(v)}$ defined in equation (b) holds at v, and thus $v \in U_{\hat{\rho}_0(v)}$. For the step from t to $t+1$, we need to distinguish two cases. If $\hat{\rho}_{t+1}(v)$ is of length 1, then it is equal to $\hat{\rho}_t(v)$, and there is nothing new to prove. Otherwise, we must consider the appropriate subformula $\varphi_{\hat{\rho}_{t+1}(v)}$ given by equation (c). We already know from the base case that the conjunct $X_{\hat{\rho}_{t+1}(v).\mathsf{first}} = X_{\hat{\rho}_0(v)}$ holds at v, with respect to any variable assignment that interprets each X_σ as U_σ. Furthermore, by the induction hypothesis, $X_{\hat{\rho}_t(u)}$ holds at every incoming neighbor u of v. Since $\{\hat{\rho}_t(u) \mid uv \in E\} \triangleright \hat{\rho}_{t+1}(v)$, we conclude that the second conjunct of $\varphi_{\hat{\rho}_{t+1}(v)}$ must also hold at v, and thus $v \in U_{\hat{\rho}_{t+1}(v)}$. Finally, assuming A accepts (G, v), we know by definition that $\hat{\rho}_t(v).\mathsf{last} \in F$ for some $t \in \mathbb{N}$. Since $v \in U_{\hat{\rho}_t(v)}$, this implies that the subformula φ_0 defined in equation (a) holds at v, and therefore that (G, v) satisfies φ.

For the converse direction of the equivalence, we have to overcome the difficulty that φ is more permissive than A, in the sense that a node v might lie in U_σ, and yet not be able to follow the trace σ under any timing of G. Intuitively, the reason why we still obtain an equivalence is that A cannot take advantage of all the information provided by any particular run, because it must ensure that for *all* digraphs, its acceptance behavior is independent of the timing. It turns out that even if v cannot traverse σ, some other node v' in an indistinguishable digraph will be able to do so. More precisely, we will show that

> if $v \in U_\sigma$, then there exists a pointed digraph (G', v'), backward bisimilar to (G, v), and a lossless-asynchronous timing τ' of G', such that $\hat{\rho}'_t(v') = \sigma$ for some $t \in \mathbb{N}$, (∗)

where $\hat{\rho}'$ is the enriched run of A on G' timed by τ'. Now suppose that (G, v) satisfies φ. By

equation (a), this means that $v \in U_\sigma$ for some trace σ such that $\sigma.\mathsf{last} \in F$. Consequently, A accepts the pointed digraph (G', v') postulated in $(*)$, based on the claim that v' traverses σ under timing τ' and the fact that A is lossless-asynchronous. Since (G, v) and (G', v') are backward bisimilar, it follows from Proposition 5 that A also accepts (G, v).

It remains to verify $(*)$. We achieve this by computing the least fixpoint $\vec{U}$ inductively and proving the statement by induction on the sequence of approximants $(\vec{U}^0, \vec{U}^1, \dots)$. Note that we do not need to consider the limit case, since $\vec{U} = \vec{U}^n$ for some $n \in \mathbb{N}$.

The base case is trivially true because all the components of $\vec{U}^0$ are empty. Furthermore, if σ consists of a single state q, then we do not even need to argue by induction, as it is evident from equation (b) that for all $j \geq 1$, node v lies in U_q^j precisely when $\delta_0(\lambda(v)) = q$. It thus suffices to set $(G', v') = (G, v)$ and choose the timing τ' arbitrarily. Clearly, we have $\hat{\rho}'_0(v') = \delta_0(\lambda(v)) = q$ if $v \in U_q^j$.

On the other hand, if σ is of length at least 2, we must assume that statement $(*)$ holds for the components of $\vec{U}^j$ in order to prove it for U_σ^{j+1}. To this end, consider an arbitrary node $v \in U_\sigma^{j+1}$. By the first conjunct in (c) and the preceding remarks regarding the trivial cases, we know that $\delta_0(\lambda(v)) = \sigma.\mathsf{first}$ (and incidentally that $j \geq 1$). Moreover, the second conjunct ensures the existence of a (possibly empty) set of traces $\mathfrak{S}$ that satisfies $\mathfrak{S} \triangleright \sigma$ and that represents a "projection" of v's incoming neighborhood at stage j. By the latter we mean that for all $\pi \in \mathfrak{S}$, there exists $u \in V$ such that $uv \in E$ and $u \in U_\pi^j$, and conversely, for all $u \in V$ with $uv \in E$, there exists $\pi \in \mathfrak{S}$ such that $u \in U_\pi^j$.

Now, for each trace $\pi \in \mathfrak{S}$ and each incoming neighbor u of v that is contained in U_π^j, the induction hypothesis provides us with a pointed digraph $(G'_{u:\pi}, u'_\pi)$ and a corresponding timing $\tau'_{u:\pi}$, as described in $(*)$. We make $n_{u:\pi} \in \mathbb{N}$ distinct copies of each such digraph $G'_{u:\pi}$. From this, we construct $G' = (V', E', \lambda')$ by taking the disjoint union of all the $\sum n_{u:\pi}$ digraphs, and adding a single new node v' with $\lambda'(v') = \lambda(v)$, together with all the edges of the form $u'_\pi v'$ (i.e., one such edge for each copy of every u'_π). Given that every $(G'_{u:\pi}, u'_\pi)$ is backward bisimilar to (G, u), we can guarantee that the same holds for (G', v') and (G, v) by choosing the numbers of digraph copies in G' such that each incoming neighbor u of v is represented by at least one incoming neighbor of v'. That is, for every u, we require that $n_{u:\pi} \geq 1$ for some π.

Finally, we construct a suitable lossless-asynchronous timing τ' of G', which proceeds in two phases to make v' traverse σ in the corresponding enriched run $\hat{\rho}'$. In the first phase, where $0 < t \leq t_1$, node v' remains inactive, which means that every τ_t assigns 0 to v' and its incoming edges. The state of v' at time t_1 is thus still $\sigma.\mathsf{first}$. Meanwhile, in every copy of each digraph $G'_{u:\pi}$, the nodes and edges behave according to timing $\tau'_{u:\pi}$ until the respective copy of u'_π has completely traversed π, whereupon the entire subgraph becomes inactive. By choosing t_1 large enough, we make sure that the FIFO buffer on each edge of the form $u'_\pi v'$ contains precisely π at time t_1. In the second phase, which lasts from $t_1 + 1$ to t_2, the only active parts of G' are v' and its incoming edges. Since the number $n_{u:\pi}$ of copies of each digraph $G'_{u:\pi}$ can be chosen as large as required, we stipulate that for every trace $\pi \in \mathfrak{S}$, the sum of $n_{u:\pi}$ over all u exceeds the lower bound m_π that is associated with π when invoking Lemma 6 for σ and $\mathfrak{S}$. Applying that lemma, we obtain a lossless-asynchronous timing segment of the subgraph induced by v' and its incoming neighbors. This segment determines our timing τ' between $t_1 + 1$ and t_2 (the other parts of G' being inactive), and gives us $\hat{\rho}'_{t_2}(v') = \sigma$, as desired. Naturally, the remainder of τ', starting at $t_2 + 1$, can be chosen arbitrarily, so long as it satisfies the properties of a lossless-asynchronous timing.

As a closing remark, note that the pointed digraph (G', v') constructed above is very similar to the standard unraveling of (G, v) into a (possibly infinite) tree. (The set of nodes

of that tree-unraveling is precisely the set of all directed paths in G that start at v; see, e.g., [1, Def. 4.51] or [2, § 3.2]). However, there are a few differences: First, we do the unraveling backwards, because we want to generate a backward bisimilar structure, where all the edges point toward the root. Second, we may duplicate the incoming neighbors (i.e., children) of each node in the tree, in order to satisfy the lower bounds imposed by Lemma 6. Third, we stop the unraveling process at a finite depth (not necessarily the same for each subtree), and place a copy of the original digraph G at every leaf. ◀

Acknowledgments. I would like to thank Olivier Carton, my PhD supervisor, for many pleasant discussions and constructive comments.

References

1. Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2002. doi:10.1017/CBO9781107050884.
2. Patrick Blackburn and Johan van Benthem. Modal logic: a semantic perspective. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 1–84. Elsevier, 2007. doi:10.1016/S1570-2464(07)80004-8.
3. Julian Bradfield and Colin Stirling. Modal mu-calculi. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 721–756. Elsevier, 2007. doi:10.1016/S1570-2464(07)80015-2.
4. Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2007. doi:10.1007/3-540-68804-8.
5. Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempiäinen, Kerkko Luosto, Jukka Suomela, and Jonni Virtema. Weak models of distributed computing, with connections to modal logic. In Darek Kowalski and Alessandro Panconesi, editors, *ACM Symposium on Principles of Distributed Computing, PODC'12, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 185–194. ACM, 2012. doi:10.1145/2332432.2332466.
6. Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempiäinen, Kerkko Luosto, Jukka Suomela, and Jonni Virtema. Weak models of distributed computing, with connections to modal logic. *Distributed Computing*, 28(1):31–53, 2015. doi:10.1007/s00446-013-0202-3.
7. Antti Kuusisto. Modal logic and distributed message passing automata. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 452–468. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.452.
8. Antti Kuusisto. Infinite networks, halting and local algorithms. In Adriano Peron and Carla Piazza, editors, *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014*, volume 161 of *EPTCS*, pages 147–160, 2014. doi:10.4204/EPTCS.161.14.
9. Giacomo Lenzi. The modal μ-calculus: a survey. *TASK Quarterly – Scientific Bulletin of the Academic Computer Centre in Gdansk*, 9(3):293–316, 2005. URL: http://task.gda.pl/quart/05-3.html.
10. Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

11 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*, volume 5 of *SIAM Monographs on Discrete Mathematics and Applications*. Society for Industrial and Applied Mathematics (SIAM), 2000. doi:10.1137/1.9780898719772.

12 Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, 2013. doi:10.1145/2431211.2431223.

A Counterexample to Thiagarajan's Conjecture on Regular Event Structures*

Jérémie Chalopin[1] and Victor Chepoi[1]

1 **LIF, CNRS and Aix-Marseille Université, Marseille, France**
`jeremie.chalopin@lif.univ-mrs.fr`
2 **LIF, CNRS and Aix-Marseille Université, Marseille, France**
`victor.chepoi@lif.univ-mrs.fr`

Abstract

We provide a counterexample to a conjecture by Thiagarajan (1996 and 2002) that regular prime event structures correspond exactly to those obtained as unfoldings of finite 1-safe Petri nets. The same counterexample is used to disprove a closely related conjecture by Badouel, Darondeau, and Raoult (1999) that domains of regular event structures with bounded $\natural$-cliques are recognizable by finite trace automata. Event structures, trace automata, and Petri nets are fundamental models in concurrency theory. There exist nice interpretations of these structures as combinatorial and geometric objects and both conjectures can be reformulated in this framework. Namely, the domains of prime event structures correspond exactly to pointed median graphs; from a geometric point of view, these domains are in bijection with pointed CAT(0) cube complexes.

A necessary condition for both conjectures to be true is that domains of respective regular event structures admit a regular nice labeling. To disprove these conjectures, we describe a regular event domain (with bounded $\natural$-cliques) that does not admit a regular nice labeling. Our counterexample is derived from an example by Wise (1996 and 2007) of a nonpositively curved square complex $\mathbf{X}$ whose universal cover $\widetilde{\mathbf{X}}$ is a CAT(0) square complex containing a particular plane with an aperiodic tiling.

1998 ACM Subject Classification F.1.1 Models of Computation, G.2.2 Graph Theory

Keywords and phrases Discrete event structures, Trace automata, Median graphs and CAT(0) cube Complexes, Unfoldings and universal covers

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.101

1 Introduction

Event structures, introduced by Nielsen, Plotkin, and Winskel [18, 29, 30], are a widely recognized abstract model of concurrent computation. An event structure is a partially ordered set of events together with a conflict relation. The partial order captures the causal dependency of events. The conflict relation models incompatibility of events so that two events that are in conflict cannot simultaneously occur in any state of the computation. Consequently, two events that are neither ordered nor in conflict may occur concurrently. The domain of an event structure consists of all computation states, called configurations. Each computation state is a subset of events subject to the constraints that no two conflicting events can occur together in the same computation and if an event occurred in a computation then all events on which it causally depends have occurred too. Therefore, the domain of an event structure $\mathcal{E}$ is the set $\mathcal{D}(\mathcal{E})$ of all finite configurations ordered by inclusion. An event e

* A full version of the paper is available at `https://arxiv.org/abs/1605.08288`.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 101; pp. 101:1–101:14


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

is said to be enabled by a configuration c if $e \notin c$ and $c \cup \{e\}$ is a configuration. The degree of an event structure $\mathcal{E}$ is the maximum number of events enabled by a configuration of $\mathcal{E}$. The future of a configuration c is the set of all finite configurations c' containing c.

Among other things, the importance of event structures stems from the fact that several fundamental models of concurrent computation lead to event structures. Nielsen, Plotkin, and Winskel [18] proved that every 1-safe Petri net N unfolds into an event structure $\mathcal{E}_N$. Later results of [19] and [30] show in fact that 1-safe Petri nets and event structures represent each other in a strong sense. In the same vein, Stark [25] established that the domains of configurations of trace automata are exactly the conflict event domains; a presentation of domains of event structures as trace monoids (Mazurkiewicz traces) or as asynchronous transition systems was given in [22] and [6], respectively. In both cases, the events of the resulting event structure are labeled in a such a way that any two events enabled by the same configuration are labeled differently (such a labeling is usually called a nice labeling). To deal with *finite* 1-safe Petri nets, Thiagarajan [26, 27] introduced the notions of regular event structure and regular trace event structure. A regular event structure $\mathcal{E}$ is an event structure with a finite number of isomorphism types of futures of configurations and finite degree. A regular trace event structure is an event structure $\mathcal{E}$ whose events can be nicely labeled by the letters of a finite trace alphabet $M = (\Sigma, I)$ in a such a way that any two concurrent events define a pair of I and there exists only a finite number of isomorphism types of labeled futures of configurations. These definitions were motivated by the fact that the event structures $\mathcal{E}_N$ arising from *finite* 1-safe Petri nets N are regular: Thiagarajan [26] proved that event structures of *finite* 1-safe Petri nets correspond to regular trace event structures. This lead Thiagarajan to formulate the following conjecture:

▶ **Conjecture 1** ([26, 27]). *An event structure $\mathcal{E}$ is isomorphic to the event structure $\mathcal{E}_N$ arising from a finite 1-safe Petri net N if and only if $\mathcal{E}$ is regular.*

Badouel, Darondeau, and Raoult [2] formulated two similar conjectures about conflict event domain that are recognizable by finite trace automata. The first one is equivalent to Conjecture 1, while the second one is formulated in a more general setting with an extra condition. We formulate their second conjecture in the particular case of event structures:

▶ **Conjecture 2** ([2]). *The domain of an event structure $\mathcal{E}$ is recognizable if and only if $\mathcal{E}$ is regular and has bounded $\natural$-cliques.*

In view of previous results, to establish Conjecture 1, it is necessary for a regular event structure $\mathcal{E}$ to define a regular nice labeling with letters from some trace alphabet (Σ, I). Nielsen and Thiagarajan [20] proved in a technically involved but very nice combinatorial way that all regular conflict-free event structures satisfy Conjecture 1. In a equally difficult and technical proof, Badouel et al. [2] proved that their conjectures hold for context-free event domains, i.e., for domains whose underlying graph is a context-free graph sensu Müller and Schupp [17]. In this paper, we present a counterexample to Thiagarajan's Conjecture based on a more geometric and combinatorial view on event structures. We show that our example also provides a counterexample to Conjecture 2 of Badouel et al.

We use the striking bijections between the domains of event structures, median graphs, and CAT(0) cube complexes. Median graphs have many nice properties and admit numerous characterizations. They have been investigated in several contexts for more than half a century, and play a central role in metric graph theory; for more detailed information, the interested reader can consult the surveys [3, 4]. On the other hand, CAT(0) cube complexes are central objects in geometric group theory [23, 24, 33]. They have been characterized in a

nice combinatorial way by Gromov [12] as simply connected cube complexes in which the links of 0-cubes are simplicial flag complexes. It was proven in [9, 21] that 1-skeleta of CAT(0) cube complexes are exactly the median graphs. Barthélemy and Constantin [5] proved that the Hasse diagrams of domains of event structures are median graphs and every pointed median graph is the domain of an event structure. The bijection between pointed median graphs and event domains established in [5] can be viewed as the classical characterization of prime event domains as prime algebraic coherent partial orders provided by Nielsen, Plotkin, and Winskel [18]. Via these bijections, the events of an event structure $\mathcal{E}$ correspond to the parallelism classes of edges of the domain $D(\mathcal{E})$ viewed as a median graph.

Our counter-example is based on Wise's [31, 32] nonpositively curved square complex $\mathbf{X}$ with one vertex and six squares, whose edges are colored in five colors, and whose colored universal cover $\widetilde{\mathbf{X}}$ contains a particular plane with an aperiodic tiling. As a result, $\widetilde{\mathbf{X}}$ is a CAT(0) square complex whose edges are colored by the colors of their images in $\mathbf{X}$ and are directed in such a way that all edges in the same parallelism class are oriented in the same way. With respect to this orientation, all vertices of $\widetilde{\mathbf{X}}$ are equivalent up to automorphism. We modify the complex $\mathbf{X}$ by taking its barycentric subdivision and by adding to the middles of the edges of $\mathbf{X}$ directed paths of five different lengths in order to encode the colors of the edges of $\mathbf{X}$ (and $\widetilde{\mathbf{X}}$) and to obtain a nonpositively curved square complex W. The universal cover $\widetilde{W}$ of W is a directed (but no longer colored) CAT(0) square complex. Since $\widetilde{W}$ is the universal cover of a finite complex W, $\widetilde{W}$ has a finite number of equivalence classes of vertices up to automorphism. From $\widetilde{W}$ we derive a domain of a regular event structure $\widetilde{W}_{\tilde{v}}$ by considering the future of an arbitrary vertex $\tilde{v}$ of $\widetilde{\mathbf{X}}$. Using the fact that $\widetilde{\mathbf{X}}$ contains a particular plane with an aperiodic tiling, we prove that $\widetilde{W}_{\tilde{v}}$ does not admit a regular nice labeling, thus $\widetilde{W}_{\tilde{v}}$ does not have a regular trace labeling.

Due to space limitations, some proofs are omitted; a full version of the paper is available on arXiv [8].

2 Event structures

2.1 Event structures and domains

An *event structure* is a triple $\mathcal{E} = (E, \leq, \#)$, where

- E is a set of *events*,
- $\leq \subseteq E \times E$ is a partial order of *causal dependency*,
- $\# \subseteq E \times E$ is a binary, irreflexive, symmetric relation of *conflict*,
- $\downarrow e := \{e' \in E : e' \leq e\}$ is finite for any $e \in E$,
- $e\#e'$ and $e' \leq e''$ imply $e\#e''$.

What we call here an event structure is usually called a *prime event structure*. Two events e', e'' are *concurrent* (notation $e' \| e''$) if they are order-incomparable and they are not in conflict. The conflict $e'\#e''$ between two elements e' and e'' is said to be *minimal* (notation, $e'\#_\mu e''$) if there is no event $e \neq e', e''$ such that either $e \leq e'$ and $e\#e''$ or $e \leq e''$ and $e\#e'$. Also define the binary relation $\lessdot \subseteq E \times E$ as follows: set $e \lessdot e'$ if and only if $e \leq e', e \neq e'$, and for every e'' if $e \leq e'' \leq e'$, then $e'' = e$ or $e'' = e'$. Given two event structures $\mathcal{E}_1 = (E_1, \leq_1, \#_1)$ and $\mathcal{E}_2 = (E_2, \leq_2, \#_2)$, a map $f : E_1 \to E_2$ is an isomorphism if f is a bijection such that $e \leq_1 e'$ iff $f(e) \leq_2 f(e')$ and $e\#_1 e'$ iff $f(e)\#_2 f(e')$ for every $e, e' \in E_1$. If such an isomorphism exists, then $\mathcal{E}_1$ and $\mathcal{E}_2$ are said to be isomorphic; notation $\mathcal{E}_1 \equiv \mathcal{E}_2$.

A *labeled event structure* $\mathcal{E}^\lambda = (\mathcal{E}, \lambda)$ is defined by an *underlying event structure* $\mathcal{E} = (E, \leq, \#)$ and a *labeling* λ that is a map from E to some alphabet Σ. Two labeled event structures $\mathcal{E}_1^{\lambda_1} = (\mathcal{E}_1, \lambda_1)$ and $\mathcal{E}_2^{\lambda_1} = (\mathcal{E}_2, \lambda_2)$ are isomorphic (notation $\mathcal{E}_1^{\lambda_1} \equiv \mathcal{E}_2^{\lambda_2}$) if there

exists an isomorphism f between the underlying event structures $\mathcal{E}_1$ and $\mathcal{E}_2$ such that $\lambda_2(f(e_1)) = \lambda_1(e_1)$ for every $e_1 \in E_1$.

A *configuration* of an event structure $\mathcal{E} = (E, \leq, \#)$ is any finite subset $c \subset E$ of events which is *conflict-free* ($e, e' \in c$ implies that e, e' are not in conflict) and *downward-closed* ($e \in c$ and $e' \leq e$ implies that $e' \in c$) [30]. Notice that $\varnothing$ is always a configuration and that $\downarrow e$ and $\downarrow e \setminus \{e\}$ are configurations for any $e \in E$. The *domain* of an event structure is the set $\mathcal{D} := \mathcal{D}(\mathcal{E})$ of all configurations of $\mathcal{E}$ ordered by inclusion; (c', c) is a (directed) edge of the Hasse diagram of the poset $(\mathcal{D}(\mathcal{E}), \subseteq)$ if and only if $c = c' \cup \{e\}$ for an event $e \in E \setminus c$. An event e is said to be *enabled* by a configuration c if $e \notin c$ and $c \cup \{e\}$ is a configuration. Denote by $en(c)$ the set of all events enabled at the configuration c. Two events are called *co-initial* if they are both enabled at some configuration c. Note that if e and e' are co-initial, then either $e \#_\mu e'$ or $e \| e'$. It is easy to see that two events e and e' are in minimal conflict $e \#_\mu e'$ if and only if $e \# e'$ and e and e' are co-initial. The *degree* $\deg(\mathcal{E})$ of an event structure $\mathcal{E}$ is the least positive integer d such that $|en(c)| \leq d$ for any configuration c of $\mathcal{E}$. We say that $\mathcal{E}$ has *finite degree* if $\deg(\mathcal{E})$ is finite. The *future* (or the *filter*) $\mathcal{F}(c)$ of a configuration c is the set of all configurations c' containing c: $\mathcal{F}(c) = \uparrow c := \{c' \in \mathcal{D}(\mathcal{E}) : c \subseteq c'\}$, i.e., $\mathcal{F}(c)$ is the principal filter of c in the ordered set $(\mathcal{D}(\mathcal{E}), \subseteq)$.

For an event structure $\mathcal{E} = (E, \leq, \natural)$, let $\natural$ be the least irreflexive and symmetric relation on the set of events E such that $e_1 \natural e_2$ if (1) $e_1 \| e_2$, or (2) $e_1 \#_\mu e_2$, or (3) there exists an event e_3 that is co-initial with e_1 and e_2 at two different configurations such that $e_1 \| e_3$ and $e_2 \#_\mu e_3$. If $e_1 \natural e_2$ and this comes from condition (3), then we write $e_1 \natural_{(3)} e_2$. A $\natural$-*clique* is a subset S of events such that $e_1 \natural e_2$ for any $e_1, e_2 \in S$.

A labeling $\lambda : E \to \Sigma$ of an event structure $\mathcal{E}$ (or of its domain $\mathcal{D}(\mathcal{E})$) is called a *nice labeling* if any two events that are co-initial have different labels [22]. A nice labeling of $\mathcal{E}$ can be reformulated as a coloring of the directed edges of the Hasse diagram of its domain $\mathcal{D}(\mathcal{E})$ subject to the following local conditions:

- **Determinism:** The edges outgoing from the same vertex of $\mathcal{D}(\mathcal{E})$ have different colors.
- **Concurrency:** the opposite edges of each square of $\mathcal{D}(\mathcal{E})$ are colored with the same color.

2.2 Regular event structures

In this subsection, we recall the definitions of regular event structures, regular trace event structures, and regular nice labelings of event structures. We closely follow the definitions and notations of [26, 27, 20]. Let $\mathcal{E} = (E, \leq, \#)$ be an event structure. Let c be a configuration of $\mathcal{E}$. Set $\#(c) = \{e' : \exists e \in c, e \# e'\}$. The *event structure rooted at* c is defined to be the triple $\mathcal{E}\backslash c = (E', \leq', \#')$, where $E' = E \setminus (c \cup \#(c))$, $\leq'$ is $\leq$ restricted to $E' \times E'$, and $\#'$ is $\#$ restricted to $E' \times E'$. It can be easily seen that the domain $\mathcal{D}(\mathcal{E}\backslash c)$ of the event structure $\mathcal{E}\backslash c$ is isomorphic to the filter $\mathcal{F}(c)$ of c in $\mathcal{D}(\mathcal{E})$ such that any configuration c' of $\mathcal{D}(\mathcal{E})$ corresponds to the configuration $c' \setminus c$ of $\mathcal{D}(\mathcal{E}\backslash c)$.

For an event structure $\mathcal{E} = (E, \leq, \#)$, define the equivalence relation $R_\mathcal{E}$ on its configurations in the following way: for two configurations c and c' set $c R_\mathcal{E} c'$ if and only if $\mathcal{E}\backslash c \equiv \mathcal{E}\backslash c'$. The *index* of an event structure $\mathcal{E}$ is the number of equivalence classes of $R_\mathcal{E}$, i.e., the number of isomorphism types of futures of configurations of $\mathcal{E}$. The event structure $\mathcal{E}$ is *regular* [26, 27, 20] if $\mathcal{E}$ has finite index and finite degree.

Now, let $\mathcal{E}^\lambda = (\mathcal{E}, \lambda)$ be a labeled event structure. For any configuration c of $\mathcal{E}$, if we restrict λ to $\mathcal{E}\backslash c$, then we obtain a labeled event structure $(\mathcal{E}\backslash c, \lambda)$ denoted by $\mathcal{E}^\lambda\backslash c$. Analogously, define the equivalence relation $R_{\mathcal{E}^\lambda}$ on its configurations by setting $c R_{\mathcal{E}^\lambda} c'$ if and only if $\mathcal{E}^\lambda\backslash c \equiv \mathcal{E}^\lambda\backslash c'$. The index of $\mathcal{E}^\lambda$ is the number of equivalence classes of $R_{\mathcal{E}^\lambda}$. We

say that an event structure $\mathcal{E}$ admits a *regular nice labeling* if there exists a nice labeling λ of $\mathcal{E}$ with a finite alphabet Σ such that $\mathcal{E}^\lambda$ has finite index.

We continue by recalling the definition of regular trace event structures from [26, 27]. A *(Mazurkiewicz) trace alphabet* is a pair $M = (\Sigma, I)$, where Σ is a finite non-empty alphabet set and $I \subset \Sigma \times \Sigma$ is an irreflexive and symmetric relation called the *independence relation*. As usual, Σ^* is the set of finite words with letters in Σ. The independence relation I induces the equivalence relation $\sim_I$, which is the reflexive and transitive closure of the binary relation $\leftrightarrow_I$: *if* $\sigma, \sigma' \in \Sigma^*$ *and* $(a, b) \in I$, *then* $\sigma ab\sigma' \leftrightarrow_I \sigma ba\sigma'$. The relation $D := (\Sigma \times \Sigma) \setminus I$ is called the *dependence relation*. An *M-labeled event structure* is a labeled event structure $\mathcal{E}^\lambda = (\mathcal{E}, \lambda)$, where $\mathcal{E} = (E, \leq, \#)$ is an event structure and $\lambda : E \to \Sigma$ is a labeling function which satisfies the following conditions:

- (LES1) $e \#_\mu e'$ implies $\lambda(e) \neq \lambda(e')$,
- (LES2) $e \lessdot e'$ or $e \#_\mu e'$, then $(\lambda(e), \lambda(e')) \in D$,
- (LES3) if $(\lambda(e), \lambda(e')) \in D$, then $e \leq e'$ or $e' \leq e$ or $e \# e'$.

We call λ a *trace labeling* of $\mathcal{E}$. The conditions (LES2) and (LES3) on the labeling function ensures that the concurrency relation $\|$ of $\mathcal{E}$ respects the independence relation I of M. In particular, since I is irreflexive, from (LES3) it follows that any two concurrent events are labeled differently. Since by (LES1) two events in minimal conflict are also labeled differently, this implies that λ is a finite nice labeling of $\mathcal{E}$.

An M-labeled event structure $\mathcal{E}^\lambda = (\mathcal{E}, \lambda)$ is *regular* if $\mathcal{E}^\lambda$ has finite index. Finally, an event structure $\mathcal{E}$ is called a *regular trace event structure* [26, 27] iff there exists a trace alphabet $M = (\Sigma, I)$ and a regular M-labeled event structure $\mathcal{E}^\lambda$ such that $\mathcal{E}$ is isomorphic to the underlying event structure of $\mathcal{E}^\lambda$. From the definition immediately follows that every regular trace event structure is also a regular event structure. It turns out that the converse is equivalent to Conjecture 1. Namely, [27] establishes the following equivalence (this result dispenses us from giving a formal definition of 1-safe Petri nets; the interested readers can find it in the papers [27, 20]):

▶ **Theorem 3** ([27, Theorem 1]). *$\mathcal{E}$ is a regular trace event structure if and only if there exists a finite 1-safe Petri net N such that $\mathcal{E}$ and $\mathcal{E}_N$ are isomorphic.*

In view of this theorem, Conjecture 1 is equivalent to the following conjecture:

▶ **Conjecture 4.** *$\mathcal{E}$ is a regular event structure iff $\mathcal{E}$ is a regular trace event structure.*

Badouel et al. [2] considered recognizable conflict event domains that are more general than the domains of event structures we consider in this paper. Since the domain of an event structure $\mathcal{E}$ is recognizable if and only if $\mathcal{E}$ is a regular trace event structure (see [16, Section 5]), Conjecture 2 can be reformulated as follows:

▶ **Conjecture 5.** *$\mathcal{E}$ is a regular event structure iff $\mathcal{E}$ is a regular trace event structure and $\mathcal{E}$ has bounded $\natural$-cliques.*

Since any regular trace labeling is a regular nice labeling, any regular event structure $\mathcal{E}$ not admitting a regular nice labeling is a counter-example to Conjecture 4 (and thus to Conjecture 1). If, additionally, $\mathcal{E}$ has bounded $\natural$-cliques, $\mathcal{E}$ is also a counter-example to Conjecture 5 (and thus to Conjecture 2).

3 Domains, median graphs, and CAT(0) cube complexes

In this section, we recall the bijections between domains of event structures and median graphs/CAT(0) cube complexes established in [1] and [5], and between median graphs and 1-skeleta of CAT(0) cube complexes established in [9] and [21].

Let $G = (V, E)$ be a simple, connected, not necessarily finite graph. The *distance* $d_G(u, v)$ between two vertices u and v is the length of a shortest (u, v)-path, and the *interval* $I(u, v)$ between u and v consists of all vertices on shortest (u, v)–paths. A graph G is *median* if for any three vertices x, y, z of G, there exists a unique vertex $m = m(x, y, z)$, called the *median* of x, y, z, simultaneously lying on the intervals $I(x, y), I(x, z)$, and $I(y, z)$. Basic examples of median graphs are trees, hypercubes, rectangular grids, and Hasse diagrams of distributive lattices and of median semilattices [3]. With any vertex v of a median graph $G = (V, E)$ is associated a *canonical partial order* $\leq_v$ defined by setting $x \leq_v y$ if and only if $x \in I(v, y)$; v is called the *basepoint* of $\leq_v$. Since G is bipartite, the Hasse diagram G_v of the partial order $(V, \leq_v)$ is the graph G in which any edge xy is directed from x to y if and only if the inequality $d_G(x, v) < d_G(y, v)$ holds. We call G_v a *pointed median graph*.

Median graphs can be obtained from hypercubes by amalgams and median graphs are themselves isometric subgraphs of hypercubes. The canonical isometric embedding of a median graph G into a (smallest) hypercube can be determined by the so called *Djoković-Winkler ("parallelism")* relation Θ on the edges of G [11, 28]. For median graphs, the equivalence relation Θ can be defined as follows. First say that two edges uv and xy are in relation Θ_0 if they are either equal or opposite edges of a 4-cycle $uvxy$ in G. Then let Θ be the transitive closure of Θ_0. We denote by $\{\Theta_i : i \in I\}$ the equivalence classes of the relation Θ (in [5], they were called parallelism classes). Each equivalence class $\Theta_i, i \in I$, is a cutset of G: namely, it splits $V(G)$ in two convex subgraphs A_i, B_i of G (A subgraph H of G is *convex* if for all $u, v \in V(H)$, $I(u, v) \subseteq V(H)$). The equivalence relation Θ is fundamental in the bijection between event structures and median graphs:

▶ **Theorem 6** ([5]). *The Hasse diagram of the domain $(\mathcal{D}(\mathcal{E}), \subseteq)$ of any event structure $\mathcal{E} = (E, \leq, \#)$ is a median graph. Conversely, for any median graph G and any basepoint v of G, the pointed median graph G_v is isomorphic to the Hasse diagram of the domain of an event structure.*

In the construction of an event structure $\mathcal{E}_v$ from a median graph G pointed at a vertex v, the events $e_i, i \in I$ of $\mathcal{E}_v$ correspond to the equivalence classes $\Theta_i, i \in I$ of Θ. Two classes Θ_i and Θ_j define concurrent events if and only if they cross, i.e., there exists a square $uvxy$ where $uv, xy \in \Theta_i$ and $uy, vx \in \Theta_j$. For two events e_i, e_j, we have $e_i \prec e_j$ if and only if the cutset Θ_i separates v from the edges of Θ_j. Finally, two events e_i, e_j are in conflict if and only if Θ_i and Θ_j do not cross and neither separates the other from v.

A *cube complex* is a cell complex X whose cells are unit Euclidean cubes of various dimensions such that any two intersecting cubes of X intersect in a common face. The 0-cubes and the 1-cubes of X are called *vertices* and *edges* of X and define the graph $X^{(1)}$, the 1-*skeleton* of X. The *star* $\mathrm{St}(v, X)$ of a vertex v of X is the subcomplex spanned by all cubes containing v. A cube complex X is *simply connected* if every cycle C of its 1-skeleton is null-homotopic, i.e., it can be contracted to a single point by elementary homotopies. Given two cube complexes X and Y, a *covering (map)* is a surjection $p\colon Y \to X$ mapping cubes to cubes and such that $p|_{\mathrm{St}(v,Y)}\colon \mathrm{St}(v, Y) \to \mathrm{St}(p(v), X)$ is an isomorphism for every vertex v in Y. The space Y is then called a *covering space* of X. A *universal cover* of X is a simply connected covering space; it always exists and it is unique up to isomorphism [13, Sections 1.3 and 4.1]. The universal cover of a complex X will be denoted by $\widetilde{X}$. In particular, if X is simply connected, then its universal cover $\widetilde{X}$ is X itself.

An important class of cube complexes studied in geometric group theory and combinatorics is the class of CAT(0) cube complexes. In this case, being CAT(0) is equivalent to the unicity of geodesics in the ℓ_2 metric; see [7] for this and other properties of CAT(0) spaces. Gromov [12] gave a beautiful combinatorial characterization of CAT(0) cube complexes as

simply connected cube complexes satisfying the following condition: *if three $(k+2)$-cubes pairwise intersect in a $(k+1)$-cube and all three intersect in a k-cube, then they are included in a $(k+3)$-cube.* A cube complex X satisfying this combinatorial condition is called a *nonpositively curved (NPC)* complex. As a corollary of Gromov's result, for any NPC complex X, its universal cover $\widetilde{X}$ is CAT(0).

There is a well-known bijection between median graphs and CAT(0) cube complexes [9, 21]. Each median graph G gives rise to a cube complex $X(G)$ obtained by replacing all hypercubes of G by Euclidean unit cubes. Endowed with the intrisic ℓ_2-metric, $X(G)$ is a CAT(0) space. Conversely, the 1-skeleton of any CAT(0) cube complex is a median graph. In fact, a graph G is median if and only if its cube complex is simply connected and G satisfies the 3-cube condition [9]: *if three squares of G pairwise intersect in an edge and all three intersect in a vertex, then they belong to a 3-cube.*

This link between event domains, median graphs, and CAT(0) cube complexes allows a more geometric and combinatorial approach to several questions on event structures (and to work only with CAT(0) cube complexes viewed as event domains). For example, this allowed [10] to disprove the so-called *nice labeling conjecture* of Rozoy and Thiagarajan [22] asserting that any event structure of finite degree admits a finite nice labeling.

4 Directed NPC Complexes

Since we can define event structures from their domains, universal covers of NPC complexes represent a rich source of event structures. To obtain regular event structures, it is natural to consider universal covers of finite NPC complexes. Moreover, since domains of event structures are directed, it is natural to consider universal covers of NPC complexes whose edges are directed. However, the resulting directed universal covers are not in general domains of event structures. In particular, the domains corresponding to pointed median graphs given by Theorem 6 cannot be obtained in this way. In order to overcome this difficulty, we introduce directed median graphs and directed NPC complexes. Using these notions, one can naturally define regular event structures starting from finite directed NPC complexes.

A *directed median graph* is a pair (G, o), where G is a median graph and o is an orientation of the edges of G in a such a way that opposite edges of squares of G have the same direction. By transitivity of Θ, all edges from the same parallelism class Θ_i of G have the same direction. Since each Θ_i partitions G into two parts, o defines a partial order $\prec_o$ on the vertex-set of G. For a vertex v of G, let $\mathcal{F}_o(v, G) = \{x \in V : v \prec_o x\}$ be the principal filter of v in the partial order $(V(G), \prec_o)$.

The following lemma shows that choosing an arbitrary vertex in a directed median graph as a basepoint, one can define the domain of an event structure.

▶ **Lemma 7.** *For any vertex v of a directed median graph (G, o), the following holds:*

1. *$\mathcal{F}_o(v, G)$ induces a convex subgraph of G;*
2. *the restriction of the partial order $\prec_o$ on $\mathcal{F}_o(v, G)$ coincides with the restriction of the canonical basepoint order $\leq_v$ on $\mathcal{F}_o(v, G)$;*
3. *$\mathcal{F}_o(v, G)$ with $\prec_o$ is the domain of an event structure;*
4. *for any vertex $u \in \mathcal{F}_o(v, G)$, the principal filter $\mathcal{F}_o(u, G)$ is included in $\mathcal{F}_o(v, G)$ and $\mathcal{F}_o(u, G)$ coincides with the principal filter of u with respect to the canonical basepoint order $\leq_v$ on $\mathcal{F}_o(v, G)$.*

A *directed NPC complex* is a pair (Y, o), where Y is a NPC complex and o is an orientation of the edges of Y in a such a way that the opposite edges of the same square of Y have the

same direction. The orientation o of the edges of Y induces in a natural way an orientation $\tilde{o}$ of the edges of its universal cover $\widetilde{Y}$, so that $(\widetilde{Y}, \tilde{o})$ is a directed NPC complex and $(\widetilde{Y}^{(1)}, \tilde{o})$ is a directed median graph. We now formulate the crucial regularity property of directed median graphs $(\widetilde{Y}^{(1)}, \tilde{o})$ when (Y, o) is finite.

▶ **Lemma 8.** *If (Y, o) is a finite directed NPC complex, then $(\widetilde{Y}^{(1)}, \tilde{o})$ is a directed median graph with at most $|V(Y)|$ isomorphism types of principal filters.*

Combining Lemmas 7 and 8, we obtain the following result.

▶ **Proposition 9.** *Let (Y, o) be a finite directed NPC complex. Then for any vertex $\tilde{v}$ of the universal cover $\widetilde{Y}$ of Y, the principal filter $\mathcal{F}_{\tilde{o}}(\tilde{v}, \widetilde{Y}^{(1)})$ with the partial order $\prec_{\tilde{o}}$ is the domain of a regular event structure with at most $|V(Y)|$ different isomorphism types of futures.*

A *square complex* X is a combinatorial 2-complex whose 2-cells are attached by closed combinatorial paths of length 4. Thus, one can consider each 2-cell as a square attached to the 1-skeleton $X^{(1)}$ of X. A square complex X is a VH*-complex* (*vertical-horizontal complex*) if the 1-cells (edges) of X are partitioned into two sets V and H called *vertical* and *horizontal* edges respectively, and the edges in each square alternate between edges in V and H. Notice that if X is a VH-complex, then X satisfies the Gromov's nonpositive curvature condition since no three squares may pairwise intersect on three edges with a common vertex. A VH-complex X is a *complete square complex* (CSC) [32] if any vertical edge and any horizontal edge incident to a common vertex belong to a common square of X. By [32, Theorem 3.8], if X is a complete square complex, then the universal cover $\widetilde{X}$ of X is isomorphic to the Cartesian product of two trees. By a *plane* Π in $\widetilde{X}$ we will mean a convex subcomplex of $\widetilde{X}$ isometric to $\mathbb{R}^2$ tiled by the grid $\mathbb{Z}^2$ into unit squares.

5 Wise's event domain $\widetilde{W_{\tilde{v}}}$

In this section, we construct the domain $\widetilde{W_{\tilde{v}}}$ of a regular event structure (with bounded $\natural$-cliques) that does not admit a regular nice labelling. To do so, we start with a directed colored CSC $\mathbf{X}$ introduced by Wise [32]. In the following, we consider directed colored VH-complexes, in which each edge has an orientation and a color. Such complexes will be denoted by bold letters, like $\mathbf{X}$. Sometimes, we need to forget the colors and the orientations of the edges of these complexes. For a complex $\mathbf{X}$, we denote by X the complex obtained by forgetting the colors and the orientations of the edges of $\mathbf{X}$ (X is called the *support* of $\mathbf{X}$), and we denote by (X, o) the directed complex obtained by forgetting the colors of $\mathbf{X}$.

5.1 Wise's square complex $\mathbf{X}$ and its universal cover $\widetilde{\mathbf{X}}$

The complex $\mathbf{X}$ consists of six squares as indicated in Figure 1 (reproducing Figure 3 of [32]). Each square has two vertical and two horizontal edges. The horizontal edges are oriented from left to right and vertical edges from bottom to top. Denote this orientation of edges by o. The vertical edges of squares are colored white, grey, and black and denoted a, b, and c, respectively. The horizontal edges of squares are colored by single or double arrow, and denoted x and y, respectively. The six squares are glued together by identifying edges of the same color and respecting the directions to obtain the square complex $\mathbf{X}$. Note that $\mathbf{X}$ has a unique vertex, five edges, and six squares. It can be directly checked that $\mathbf{X}$ is a complete square complex, and consequently (X, o) is a directed NPC complex. Let H_X denote the subcomplex of X consisting of the 2 horizontal edges and let V_X denote the subcomplex of X consisting of the 3 vertical edges.

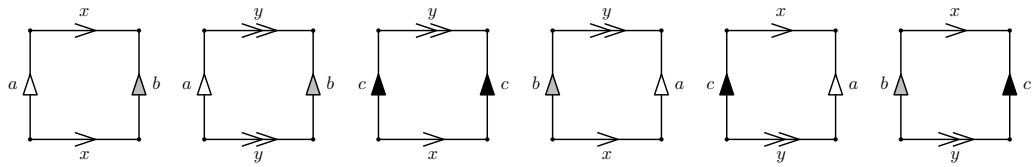


Figure 1 The 6 squares defining the complex $\mathbf{X}$.

The universal cover $\widetilde{H_X}$ of H_X is the 4-regular infinite tree F_4. Its edges inherit the orientations from their images in H_X: each vertex of $\widetilde{H_X}$ has two incoming and two outgoing arcs. Analogously, the universal cover $\widetilde{H_V}$ of H_V is the 6-regular infinite tree F_6 where each vertex has three incoming and three outgoing arcs. Let $\tilde{v}_1$ be any vertex of $\widetilde{H_X}$. Then the principal filter of $\tilde{v}_1$ is the infinite binary tree T_2 rooted at $\tilde{v}_1$: all its vertices except $\tilde{v}_1$ have one incoming and two outgoing arcs, while $\tilde{v}_1$ has two outgoing arcs and no incoming arc. Analogously, the principal filter of any vertex $\tilde{v}_2$ in the ordered set $\widetilde{H_V}$ is the infinite ternary tree T_3 rooted at $\tilde{v}_2$.

Let $\widetilde{\mathbf{X}}$ be the universal cover of $\mathbf{X}$ and let $p : \widetilde{\mathbf{X}} \to \mathbf{X}$ be a covering map. Let $\widetilde{X}$ denote the support of $\widetilde{\mathbf{X}}$. Since $\mathbf{X}$ is a CSC, by [32, Theorem 3.8], $\widetilde{X}$ is the Cartesian product $F_4 \times F_6$ of the trees F_4 and F_6. The edges of $\widetilde{\mathbf{X}}$ are colored and oriented as their images in $\mathbf{X}$, and are also classified as horizontal or vertical edges. The squares of $\widetilde{\mathbf{X}}$ are oriented as their images in $\mathbf{X}$, thus two opposite edges of the same square of $\widetilde{\mathbf{X}}$ have the same direction. This implies that all classes of parallel edges of $\widetilde{\mathbf{X}}$ are oriented in the same direction. Denote this orientation of the edges of $\widetilde{\mathbf{X}}$ by $\tilde{o}$. The 1-skeleton $\widetilde{X}^{(1)}$ of $\widetilde{X}$ together with $\tilde{o}$ is a directed median graph. Let $\tilde{v} = (\tilde{v}_1, \tilde{v}_2)$ be any vertex of $\widetilde{X}$, where $\tilde{v}_1$ and $\tilde{v}_2$ are the coordinates of $\tilde{v}$ in the trees F_4 and F_6. Then the principal filter $\mathcal{F}_{\tilde{o}}(\tilde{v}, \widetilde{X}^{(1)})$ of $\tilde{v}$ is the Cartesian product of the principal filters of $\tilde{v}_1$ in F_4 and of $\tilde{v}_2$ in F_6, i.e., is isomorphic to $T_2 \times T_3$.

By Lemma 7, the orientation of the edges of $\mathcal{F}_{\tilde{o}}(\tilde{v}, \widetilde{X}^{(1)})$ corresponds to the canonical basepoint orientation of $\mathcal{F}_{\tilde{o}}(\tilde{v}, \widetilde{X}^{(1)})$ with $\tilde{v}$ as the basepoint. Moreover, by Proposition 9, $\mathcal{F}_{\tilde{o}}(\tilde{v}, \widetilde{X}^{(1)})$ is the domain of a regular event structure with one isomorphism type of futures.

5.2 Aperiodicity of $\widetilde{\mathbf{X}}$

We recall here the main properties of $\widetilde{\mathbf{X}}$ established in [32, Section 5]. Let $\tilde{v} = (\tilde{v}_1, \tilde{v}_2)$ be an arbitrary vertex of $\widetilde{\mathbf{X}}$, where $\tilde{v}_1$ and $\tilde{v}_2$ are defined as before. From the definition of the covering map, the loop of $\mathbf{X}$ colored y gives rise to a bi-infinite horizontal path P_y of $\widetilde{\mathbf{X}}^{(1)}$ passing via $\tilde{v}$ and whose all edges are colored y and are directed from left to right. Analogously, there exists a bi-infinite vertical path P_c of $\widetilde{\mathbf{X}}^{(1)}$ passing via $\tilde{v}$ and whose all edges are colored c and are directed from bottom to top.

The projection of P_y on the horizontal factor F_4 is a bi-infinite path P^h of F_4 passing via $\tilde{v}_1$. Analogously, the projection of P_c on the vertical factor F_6 is a bi-infinite path P^v of F_6 passing via $\tilde{v}_2$. Consequently, the convex hull $\mathrm{conv}(P_y \cup P_c)$ of $P_y \cup P_c$ in the graph $\widetilde{\mathbf{X}}^{(1)}$ is isomorphic to the Cartesian product of $P^h \times P^v$ of the paths P^h and P^v. Therefore the subcomplex of $\widetilde{\mathbf{X}}$ spanned by $\mathrm{conv}(P_y \cup P_c)$ is a plane Π_{yc} tiled into squares (recall that each square is of one of 6 types and its sides are colored by the letters a, b, c, x, y), see Figure 2.

In our counterexample we will use the following result of [32] that was used to show that the plane Π_{yc} is not tiled periodically by the preimages of the squares of $\mathbf{X}$. Denote by P_y^+ the (directed) subpath of P_y having $\tilde{v}$ as the origin (this is a one-infinite horizontal path). Analogously, let P_c^+ be the (vertical) subpath of P_c having $\tilde{v}$ as the origin. The convex hull of $P_y^+ \cup P_c^+$ is a quarter of the plane Π_{yc}, which we denote by Π_{yc}^{++}. Any shortest path in

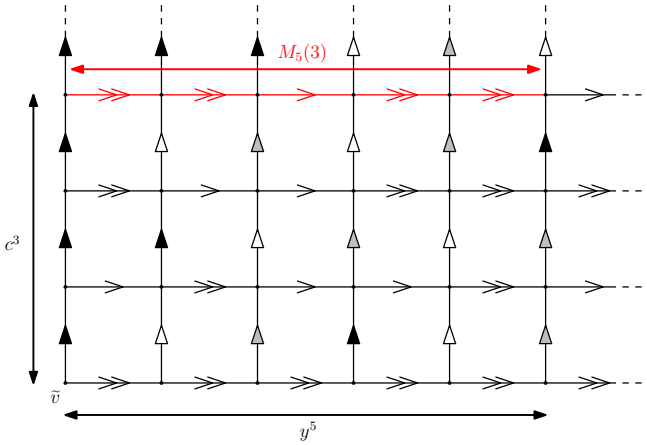


Figure 2 Part of the plane Π_{yc}^{++} appearing in $\widetilde{\mathbf{X}}$.

$\widetilde{\mathbf{X}}^{(1)}$ from $\tilde{v}$ to a vertex $\tilde{u} \in \Pi_{yc}^{++}$ can be viewed as a word in the alphabet $A = \{a, b, c, x, y\}$. For an integer $n \geq 0$, denote by y^n the horizontal subpath of P_y^+ beginning at $\tilde{v}$ and having length n. Analogously, for an integer $m \geq 0$, denote by c^m the vertical subpath of P_c^+ beginning at $\tilde{v}$ and having length m. Let $M_n(m)$ denote the horizontal path of Π_{yc}^{++} of length n beginning at the endpoint of the vertical path c^m. $M_n(m)$ determines a word which is the label of the side opposite to y^n in the rectangle which is the convex hull of y^n and c^m (see Figure 2). Let $M_n(m)$ also denote this corresponding word.

▶ **Proposition 10** ([32, Proposition 5.9]). *For each n, the words $\{M_n(m) : 0 \leq m \leq 2^n - 1\}$ are all distinct, and thus, every positive word in x and y of length n is $M_n(m)$ for some m.*

5.3 The square complex W and its universal cover $\widetilde{W}$

Let $\beta\mathbf{X}$ denote the first barycentric subdivision of $\mathbf{X}$: each square C of $\mathbf{X}$ is subdivided into four squares C_1, C_2, C_3, C_4 by adding a middle vertex to each edge of C and connecting it to the center of C by an edge. This way each edge e of C is subdivided into two edges e_1, e_2, which inherit the orientation and the color of e. The four edges connecting the middle vertices of the edges of C to the center of C are oriented from left to right and from bottom to top (see the middle figure of Figure 3). Denote the resulting orientation by o'. This way, $(\beta\mathbf{X}, o')$ is a directed and colored square complex. Again, denote by βX the support of $\beta\mathbf{X}$. The universal cover $\widetilde{\beta X}$ of βX is the Cartesian product $\beta F_4 \times \beta F_6$ of the trees βF_4 and βF_6, where βF_4 is the first barycentric subdivision of F_4 and βF_6 is the first barycentric subdivision of F_6. Additionally, $(\widetilde{\beta X}, \tilde{o}')$ is a directed CAT(0) square complex. We assign a *type* to each vertex of $\widetilde{\beta\mathbf{X}}$: the preimage of the unique vertex of $\mathbf{X}$ is of type 0 and is called a 0-*vertex*, the preimages of the middles of edges of $\mathbf{X}$ are of type 1 and are called 1-*vertices*, and the preimages of centers of squares of $\mathbf{X}$ are of type 2 and are called 2-*vertices*.

To encode the colors of the edges of $\mathbf{X}$, we introduce our central object, the square complex W (whose edges are no longer colored). Let $A = \{a, b, c, x, y\}$ and let $r : A \to \{1, 2, 3, 4, 5\}$ be a bijective map. The complex W is obtained from βX by adding to each 1-vertex z of βX a path R_z of length $r(\alpha)$ if z is the middle of an edge colored $\alpha \in A$ in $\mathbf{X}$. The path R_z has one end at z (called the *root* of R_z) and z is the unique common vertex of R_z and βX (we call such added paths R_z *tips*). Denote by o^* the orientation of the edges of W defined as follows: the edges of βX are oriented as in $(\beta X, o)$ and the edges of tips are oriented away

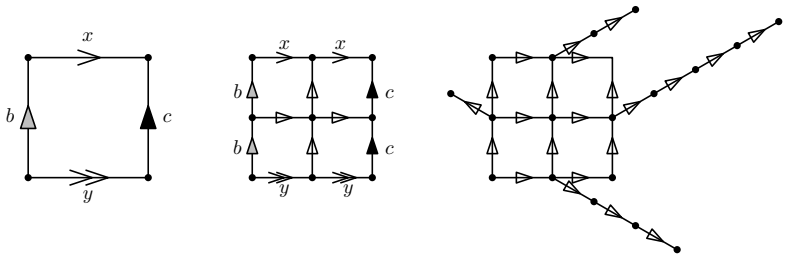


Figure 3 A square of $\mathbf{X}$ and the corresponding subcomplexes in $(\beta\mathbf{X}, o')$ and (W, o^*).

from their roots (see the rightmost figure of Figure 3 for the encoding of the last square of Figure 1). As a result, we obtain a finite directed NPC square complex (W, o^*).

Consider the universal cover $\widetilde{W}$ of W. It can be viewed as the complex $\widetilde{\beta X}$ with a path of length $r(\alpha)$ added to each 1-vertex which encodes an edge of $\widetilde{\mathbf{X}}$ of color $\alpha \in A$. We say that the vertices of $\widetilde{W}$ lying only on tips are of type 3 and they are called 3-*vertices*. Let $\tilde{o}^*$ denote the orientation of the edges of $\widetilde{W}$ induced by the orientation o^* of W. Then $(\widetilde{W}, \tilde{o}^*)$ is a directed CAT(0) square complex. Since W is finite, the directed median graph $(\widetilde{W}^{(1)}, \tilde{o}^*)$ has a finite number of isomorphisms types of principal filters $\mathcal{F}_{\tilde{o}^*}(\tilde{z}, \widetilde{W}^{(1)})$.

Let $\tilde{v}$ be any 0-vertex of $\widetilde{W}$. Denote by $\widetilde{W}_{\tilde{v}}$ the principal filter $\mathcal{F}_{\tilde{o}^*}(\tilde{v}, \widetilde{W}^{(1)})$ of vertex $\tilde{v}$ in $(\widetilde{W}^{(1)}, \prec_{\tilde{o}^*})$. By Proposition 9, $\widetilde{W}_{\tilde{v}}$ together with the partial order $\prec_{\tilde{o}^*}$ is the domain of a regular event structure, which we call *Wise's event domain*. Since vertices of different types of $\widetilde{W}$ are incident to a different number of outgoing squares, any isomorphism between two filters of $(\widetilde{W}_{\tilde{v}}, \prec_{\tilde{o}^*})$ preserves the types of vertices. We summarize all this in the following:

▶ **Proposition 11.** *$(\widetilde{W}_{\tilde{v}}, \prec_{\tilde{o}^*})$ is the domain of a regular event structure. Any isomorphism between any two filters of $(\widetilde{W}_{\tilde{v}}, \prec_{\tilde{o}^*})$ preserves the types of vertices.*

5.4 $(\widetilde{W}_{\tilde{v}}, \prec_{\tilde{o}^*})$ does not have a regular nice labeling

In this subsection we prove that the event structure associated to Wise's event domain is a counterexample to Thiagarajan's conjecture (Theorem 12) and to the conjecture of Badouel et al. [2] (Theorem 12 and Proposition 13).

▶ **Theorem 12.** *$(\widetilde{W}_{\tilde{v}}, \prec_{\tilde{o}^*})$ does not admit a regular nice labeling.*

Proof. Since $\widetilde{W}_{\tilde{v}}$ is the principal filter of a 0-vertex $\tilde{v}$, $\widetilde{W}_{\tilde{v}}$ contains all vertices of $\widetilde{\mathbf{X}}$ located in the quarter of plane Π^{++}_{yc} of $\widetilde{\mathbf{X}}$, in particular it contains the vertices of the paths P^+_c and P^+_y. Notice also that $\widetilde{W}_{\tilde{v}}$ contains the barycenters and the tips corresponding to the edges of Π^{++}_{yc}. Suppose by way of contradiction that $\widetilde{W}_{\tilde{v}}$ has a regular nice labeling λ. Since $\widetilde{W}_{\tilde{v}}$ has only a finite number of isomorphism types of labeled filters, the vertical path P^+_c contains two 0-vertices, $\tilde{z}'$ and $\tilde{z}''$, which have isomorphic labeled principal filters. Let $\tilde{z}'$ be the end of the vertical subpath c^k of P^+_c and $\tilde{z}''$ be the end of the vertical subpath c^m of P^+_c, and suppose without loss of generality that $k < m$. Let $n > 0$ be a positive integer such that $m \leq 2^n - 1$. Consider the horizontal convex paths $M_n(k)$ and $M_n(m)$ of Π^{++}_{yc} of length n beginning at the vertices $\tilde{z}'$ and $\tilde{z}''$, respectively. For any $0 \leq i \leq n$, denote by $\tilde{z}_{k,i}$ the ith vertex of $M_n(k)$ (in particular, $\tilde{z}_{k,0} = \tilde{z}'$). Analogously, denote by $\tilde{z}_{m,i}$ the ith vertex of $M_n(m)$ (in particular, $\tilde{z}_{m,0} = \tilde{z}''$). In $\widetilde{W}_{\tilde{v}}$, the paths $M_n(k)$ and $M_n(m)$ give rise to two convex horizontal paths $M^*_n(k)$ and $M^*_n(m)$ obtained from $M_n(k)$ and $M_n(m)$ by subdividing their edges. Denote by $\tilde{u}_{k,i}$ the unique common neighbor of $\tilde{z}_{k,i}$ and $\tilde{z}_{k,i+1}$, $0 \leq i < n$, in

$M_n^*(k)$ (and in $\widetilde{W}^{(1)}$). Analogously, denote by $\tilde{u}_{m,i}$ the unique common neighbor of $\tilde{z}_{m,i}$ and $\tilde{z}_{m,i+1}$, $0 \leq i < n$. The paths $M_n^*(k)$ and $M_n^*(m)$ belong to the principal filters $\mathcal{F}_{\tilde{o}^*}(\tilde{z}', \widetilde{W}^{(1)})$ and $\mathcal{F}_{\tilde{o}^*}(\tilde{z}'', \widetilde{W}^{(1)})$, respectively.

By Proposition 10, the words $M_n(k)$ and $M_n(m)$ are different. Let f be an isomorphism between the filters $\mathcal{F}_{\tilde{o}^*}(\tilde{z}_{k,0}, \widetilde{W}^{(1)})$ and $\mathcal{F}_{\tilde{o}^*}(\tilde{z}_{m,0}, \widetilde{W}^{(1)})$. Since the words $M_n(k)$ and $M_n(m)$ are different, from the choice of the lengths of tips in the complexes W and $\widetilde{W}$ it follows that f cannot map the path $M_n^*(k)$ to the path $M_n^*(m)$ by a vertical translation, i.e., there exists an index $0 \leq j < n$ such that $f(\tilde{z}_{k,j+1}) \neq \tilde{z}_{m,j+1}$; let i be the smallest such index. Set $\tilde{z} := f(\tilde{z}_{k,i+1})$ and $\tilde{u} := f(\tilde{u}_{k,i})$. Since f preserves the types of vertices, $\tilde{z}$ is a 0-vertex and $\tilde{u}$ is a 1-vertex. Since f maps a convex path $M_n^*(k)$ to a convex path, $\tilde{u}$ is the unique common neighbor of $\tilde{z}_{m,i}$ and $\tilde{z}$. Since each 1-vertex is the barycenter of a unique edge of $\widetilde{\mathbf{X}}$ and $\tilde{z} \neq \tilde{z}_{m,i+1}$, we deduce that $\tilde{u} \neq \tilde{u}_{m,i}$. The edge $\tilde{z}_{k,i}\tilde{u}_{k,i}$ is directed from $\tilde{z}_{k,i}$ to $\tilde{u}_{k,i}$. Analogously the edges $\tilde{z}_{m,i}\tilde{u}_{m,i}$ and $\tilde{z}_{m,i}\tilde{u}$ are directed from $\tilde{z}_{m,i}$ to $\tilde{u}_{m,i}$ and $\tilde{u}$, respectively. Since $\tilde{z}_{k,i}\tilde{u}_{k,i}$ and $\tilde{z}_{m,i}\tilde{u}_{m,i}$ are parallel edges, they define the same event and therefore $\lambda(\tilde{z}_{k,i}\tilde{u}_{k,i}) = \lambda(\tilde{z}_{m,i}\tilde{u}_{m,i})$. On the other hand, since f maps the edge $\tilde{z}_{k,i}\tilde{u}_{k,i}$ to the edge $\tilde{z}_{m,i}\tilde{u}$ and the map f preserves the labels, we have $\lambda(\tilde{z}_{k,i}\tilde{u}_{k,i}) = \lambda(\tilde{z}_{m,i}\tilde{u})$. As a result, $\tilde{z}_{m,i}$ has two outgoing edges, $\tilde{z}_{m,i}\tilde{u}_{m,i}$ and $\tilde{z}_{m,i}\tilde{u}$, having the same label, contrary to the assumption that λ is a nice labeling. This contradiction shows that $(\widetilde{W}_{\tilde{v}}, \prec_{\tilde{o}^*})$ does not admit a regular nice labeling. This concludes the proof of the theorem. ◀

▶ **Proposition 13.** *Wise's event domain $(\widetilde{W}_{\tilde{v}}, \prec_{\tilde{o}^*})$ has bounded $\natural$-cliques.*

6 Conclusions and open questions

In this paper, we presented an example of a regular event domain $\widetilde{W}_{\tilde{v}}$ with bounded degree and bounded $\natural$-cliques which does not admit a regular nice labeling. Consequently, the domain $\widetilde{W}_{\tilde{v}}$ is not recognizable and the prime event structure whose domain is $\widetilde{W}_{\tilde{v}}$ is not a regular trace event structure. This provides a counterexample to Conjecture 1 of Thiagarajan and Conjecture 2 of Badouel, Darondeau, and Raoult.

The event domain $\widetilde{W}_{\tilde{v}}$ is a 2-dimensional CAT(0) cube complex. The proof that our example $\widetilde{W}_{\tilde{v}}$ does not admit a regular nice labeling strongly uses the fact that the universal cover $\widetilde{\mathbf{X}}$ of Wise's complex $\mathbf{X}$ [32] contains a particular aperiodic tiled plane (that is called *antitorus* by Wise). We think that the relationship between the existence of aperiodic planes and nonexistence of regular labelings is more general. As observed by Kari and Papasoglu [14], any *4-way deterministic* tile-set gives rise to a CAT(0) VH-complex that is the universal cover of a finite NPC complex. In [14], they presented a 4-way deterministic aperiodic tile-set T_{KP}, i.e., all tilings of $\mathbb{R}^2$ using tiles from T_{KP} are aperiodic. Based on this result, Lukkarila [15] proved that for 4-way deterministic tile-sets the tiling problem is undecidable. We conjecture that one can use this result to show that *deciding if a regular event domain admits a regular nice labeling is undecidable*. As a first step in this direction, our proof can be adapted to show that any 4-way deterministic aperiodic tile-set T (in particular, T_{KP}) also provides a counterexample to Conjectures 1 and 2.

Even if Conjecture 1 does not hold in general, it would be interesting to exhibit classes of event structures for which this conjecture is true. Badouel et al. [2] showed that both conjectures hold for context-free domains. Context-free graphs are particular Gromov-hyperbolic graphs. An interesting challenge would be to *establish Conjecture 1 for Gromov-hyperbolic domains*. A positive answer would settle the previous undecidability question. As we noticed already, Conjecture 1 was positively solved by Nielsen and Thiagarajan [20] for conflict-free event structures. A possible way to extend their result is to *consider this*

conjecture for confusion-free domains introduced by Nielsen et al. [18]. From geometric and combinatorial points of view, context-free and conflict-free domains have quite different structural properties and give rise to different kinds of CAT(0) cube complexes. For instance, in context-free domains (and more generally, hyperbolic domains), isometric square-grids are bounded while conflict-free domains can contain arbitrarily large square-grids.

Acknowledgements. We are grateful to P. S. Thiagarajan for some email exchanges on Conjecture 1 and paper [20] and to our colleague R. Morin for several useful discussions.

References

1 F. Ardila, M. Owen, and S. Sullivant. Geodesics in CAT(0) cubical complexes. *Adv. Appl. Math.*, 48(1):142–163, 2012.

2 E. Badouel, Ph. Darondeau, and J.-C. Raoult. Context-free event domains are recognizable. *Inf. Comput.*, 149(2):134–172, 1999.

3 H.-J. Bandelt and V. Chepoi. Metric graph theory and geometry: a survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry: Twenty Years Later*, volume 453 of *Contemp. Math.*, pages 49–86. AMS, Providence, RI, 2008.

4 H.-J. Bandelt and J. Hedlíková. Median algebras. *Discr. Math.*, 45(1):1–30, 1983.

5 J.-P. Barthélemy and J. Constantin. Median graphs, parallelism and posets. *Discr. Math.*, 111(1-3):49–63, 1993.

6 M. A. Bednarczyk. *Categories of Asynchronous Systems*. PhD thesis, University of Sussex, 1987.

7 M. R. Bridson and A. Haefliger. *Metric Spaces of Non-Positive Curvature*, volume 319 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, Berlin, 1999.

8 J. Chalopin and V. Chepoi. A counterexample to Thiagarajan's conjecture. *arXiv preprint*, 2016. URL: `https://arxiv.org/abs/1605.08288`, `arXiv:1605.08288`.

9 V. Chepoi. Graphs of some CAT(0) complexes. *Adv. Appl. Math.*, 24(2):125–179, 2000.

10 V. Chepoi. Nice labeling problem for event structures: a counterexample. *SIAM J. Comput.*, 41(4):715–727, 2012.

11 D.Ž. Djoković. Distance-preserving subgraphs of hypercubes. *J. Comb. Theory, Ser. B*, 14(3):263–267, 1973.

12 M. Gromov. Hyperbolic groups. In S. M. Gersten, editor, *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987.

13 A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge„ 2002.

14 J. Kari and P. Papasoglu. Deterministic aperiodic tile sets. *GAFA, Geom. Funct. Anal.*, 9(2):353–369, 1999.

15 V. Lukkarila. The 4-way deterministic tiling problem is undecidable. *Theor. Comput. Sci.*, 410(16):1516–1533, 2009.

16 R. Morin. Concurrent automata vs. asynchronous systems. In *MFCS 2005*, volume 3618 of *LNCS*, pages 686–698. Springer, 2005.

17 D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.

18 M. Nielsen, G. D. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theor. Comput. Sci.*, 13:85–108, 1981.

19 M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Transition systems, event structures and unfoldings. *Inf. Comput.*, 118(2):191–207, 1995.

20 M. Nielsen and P. S. Thiagarajan. Regular event structures and finite Petri nets: the conflict-free case. In *ICATPN 2002*, volume 2360 of *LNCS*, pages 335–351. Springer, 2002.

21 M. Roller. Poc sets, median algebras and group actions. Technical report, Univ. of Southampton, 1998.

22 B. Rozoy and P. S. Thiagarajan. Event structures and trace monoids. *Theor. Comput. Sci.*, 91(2):285–313, 1991.

23 M. Sageev. Ends of group pairs and non-positively curved cube complexes. *Proc. London Math. Soc.*, s3-71(2):585–617, 1995.

24 M. Sageev. CAT(0) cube complexes and groups. In M. Bestvina, M. Sageev, and K. Vogtmann, editors, *Geometric Group Theory*, volume 21 of *IAS/Park City Mathematics Series*, pages 6–53. AMS, Institute for Advanced Study, 2012.

25 E. W. Stark. Connections between a concrete and an abstract model of concurrent systems. In *Mathematical Foundations of Programming Semantics 1989*, volume 442 of *LNCS*, pages 53–79. Springer, 1989.

26 P. S. Thiagarajan. Regular trace event structures. Technical Report BRICS RS-96-32, Computer Science Department, Aarhus University, Aarhus, Denmark, 1996.

27 P. S. Thiagarajan. Regular event structures and finite Petri nets: A conjecture. In *Formal and Natural Computing*, volume 2300 of *LNCS*, pages 244–256. Springer, 2002.

28 P. M. Winkler. Isometric embedding in products of complete graphs. *Discr. Appl. Math.*, 7(2):221–225, 1984.

29 G. Winskel. *Events in computation.* PhD thesis, Edinburgh Univ., 1980.

30 G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science (Vol. 4)*, pages 1–148. Oxford University Press, 1995.

31 D. T. Wise. *Non-positively curved squared complexes, aperiodic tilings, and non-residually finite groups.* PhD thesis, Princeton University, 1996.

32 D. T. Wise. Complete square complexes. *Comment. Math. Helv*, 82(4):683–724, 2007.

33 D. T. Wise. *From Riches to Raags: 3-manifolds, Right-angled Artin Groups, and Cubical Geometry*, volume 117 of *CBMS Regional Conference Series in Mathematics.* AMS, Providence, RI, 2012.

⋆-Liftings for Differential Privacy*†

Gilles Barthe[1], Thomas Espitau[2], Justin Hsu[3], Tetsuya Sato[4], and Pierre-Yves Strub[5]

1 **IMDEA Software Institute, Madrid, Spain**
gjbarthe@gmail.com
2 **Sorbonne Universités, UPMC Paris 6, Paris, France**
t.espitau@gmail.com
3 **University of Pennsylvania, Philadelphia, PA, USA**
email@justinh.su
4 **Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan**
satoutet@kurims.kyoto-u.ac.jp
5 **École Polytechnique, Palaiseau, France**
pierre-yves@strub.nu

Abstract

Recent developments in formal verification have identified *approximate liftings* (also known as *approximate couplings*) as a clean, compositional abstraction for proving differential privacy. There are two styles of definitions for this construction. Earlier definitions require the existence of one or more witness distributions, while a recent definition by Sato uses universal quantification over all sets of samples. These notions have different strengths and weaknesses: the universal version is more general than the existential ones, but the existential versions enjoy more precise composition principles.

We propose a novel, existential version of approximate lifting, called ⋆-*lifting*, and show that it is equivalent to Sato's construction for discrete probability measures. Our work unifies all known notions of approximate lifting, giving cleaner properties, more general constructions, and more precise composition theorems for both styles of lifting, enabling richer proofs of differential privacy. We also clarify the relation between existing definitions of approximate lifting, and generalize our constructions to approximate liftings based on f-divergences.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Differential Privacy, Probabilistic Couplings, Formal Verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.102

1 Introduction

Differential privacy [7] is a rigorous notion of statistical privacy that delivers strong individual guarantees for privacy-preserving computations. Informally, differential privacy guarantees to every individual that their (non)-participation in a database will have a small (in a rigorous, quantitative sense) effect on the results obtained by third parties when querying the database. The formal definition of differential privacy is parametrized by two non-negative real numbers, (ϵ, δ). These parameters quantify the effect of individuals on the output of the private query;

* A full version of the paper is available at https://arxiv.org/abs/1705.00133.
† This work is partially supported by a grant from the NSF (TWC-1513694) and a grant from the Simons Foundation (#360368 to Justin Hsu).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 102; pp. 102:1–102:12

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

smaller values give stronger privacy guarantees. The main strengths of differential privacy lie in its theoretical elegance, minimal assumptions, and flexibility for many applications.

Motivated by the importance of differential privacy, programming language researchers have developed approaches based on dynamic analysis, type systems, and program logics for formally proving differential privacy for programs. (We refer the interested reader to a recent survey [4] for an overview of this growing field.) In this paper, we consider approaches based on relational program logics [5, 6, 10, 2, 3, 11]. To capture the quantitative nature of differential privacy, these systems rely on a quantitative generalization of probabilistic couplings (see, e.g., [9, 13, 14]), called *approximate liftings* or (ϵ, δ)-liftings. Existing works have considered several potential definitions. While all definitions support compositional reasoning and enable program logics that can verify complex examples from the privacy literature, the various notions of approximate liftings have different strengths and weaknesses.

Broadly speaking, one class of definitions require the existence of one or two *witness distributions* that "couple' the two executions of programs. The earliest definition [5] supports accuracy-based reasoning for the Laplace mechanism, while subsequent definitions [6, 10] support more precise composition principles from differential privacy and can be generalized to other notions of distance on distributions. These definitions, and their associated program logics, were designed for discrete distributions.

In the course of extending these ideas to continuous distributions, Sato [11] proposes a radically different notion of approximate lifting, which does not rely on witness distributions. Instead, it uses a universal quantification over all sets of samples. Sato shows that this definition is strictly more general than the existential versions, but it is unclear (a) whether the gap can be closed and (b) whether his construction satisfies the same composition principles enjoyed by some existential definitions.

As a consequence, there is currently no single approximate lifting with the properties needed to support all existing formalized proofs of differential privacy. Furthermore, some of the most involved privacy proofs cannot be formalized at all, as their proofs require a combination of tools from several kinds of approximate liftings.

Outline of the paper

After reviewing the necessary mathematical preliminaries in Section 2, we introduce our main technical contribution: a new, existential definition of approximate lifting. This construction, which we call ⋆-*lifting*, is a generalization of an existing definition by Barthe and Olmedo [6, 10]. The key idea is to allow the witness distributions to have a larger domain, broadening the class of approximate liftings. By a maximum flow/minimum cut argument, we show that ⋆-liftings are equivalent to Sato's lifting over discrete distributions. This equivalence can be viewed as an approximate version of Strassen's theorem [12], a classical result in probability theory describing the existence of probabilistic couplings. We present the definition of ⋆-lifting and the proof of equivalence in Section 3.

Then, we show that ⋆-liftings satisfy desirable theoretical properties. We are able to leverage the equivalence of liftings in two ways. In one direction, Sato's definition gives simpler proofs of more general properties of ⋆-liftings. In the other direction, ⋆-liftings – like other existential definitions – can smoothly incorporate composition principles from the theory of differential privacy. Our connection shows that Sato's definition can use these principles in the discrete case. We describe the key theoretical properties of ⋆-liftings in Section 4.

Finally, we provide a thorough comparison of ⋆-lifting with existing definitions of approximate lifting in Section 5, and describe how to construct ⋆-liftings for more general version of approximate liftings based on f-divergences in Section 6.

Overall, the equivalence of $\star$-liftings and Sato's lifting, along with the natural theoretical properties satisfied by the common notion, suggest that these definitions are two views on the same concept: an approximate version of probabilistic coupling.

2 Background

To model probabilistic behavior, we work with *discrete sub-distributions*.

▶ **Definition 1.** A *sub-distribution* over a set A is defined by its mass function $\mu : A \to \mathbb{R}^+$, which gives the probability of the singleton events $a \in A$. This mass function must be s.t. $|\mu| \triangleq \sum_{a \in A} \mu(a)$ is well-defined and at most 1. In particular, the *support* $\mathrm{supp}(\mu) \triangleq \{a \in A \mid \mu(a) \neq 0\}$ must be discrete (i.e. finite or countably infinite). When the *weight* $|\mu|$ is equal to 1, we call μ a *(proper) distribution*. We let $\mathbb{D}(A)$ denote the set of sub-distributions over A. The probability of an event $E(x)$ w.r.t. μ, written $\mathbb{P}_{x\sim\mu}[E(x)]$ or $\mathbb{P}_{\mu}[E]$, is defined as $\sum_{x \in A \mid E(x)} \mu(x)$.

Simple examples of sub-distributions include the *null sub-distribution* $\mathbb{0}^A \in \mathbb{D}(A)$, which maps each element of A to 0, and the *Dirac distribution centered on* x, written $\mathbb{1}_x$, which maps x to 1 and all other elements to 0. One can equip distributions with a monadic structure using the Dirac distributions $\mathbb{1}_x$ for the unit and *distribution expectation* $\mathbb{E}_{x\sim\mu}[f(x)]$ for the bind; if μ is a distribution over A and f has type $A \to \mathbb{D}(B)$, then the bind defines a sub-distribution over B: $\mathbb{E}_{a\sim\mu}[f(a)] : b \mapsto \sum_a \mu(a) \cdot f(a)(b)$.

If $f : A \to B$, we can lift f to a function $f^\sharp : \mathbb{D}(A) \to \mathbb{D}(B)$ as follows: $f^\sharp(\mu) \triangleq \mathbb{E}_{a\sim\mu}[\mathbb{1}_{f(a)}]$ – or, equivalently, $f^\sharp(\mu) : b \mapsto \mathbb{P}_{a\sim\mu}[a \in f^{-1}(b)]$. For instance, when working with sub-distributions over pairs, this allows to obtain the probabilistic versions $\pi_1^\sharp$ and $\pi_2^\sharp$ (called *marginals*) of the usual projections π_1 and π_2. One can check that the *first* and *second marginals* $\pi_1^\sharp(\mu)$ and $\pi_2^\sharp(\mu)$ of a distribution μ over $A \times B$ are also given by the following equations: $\pi_1^\sharp(\mu)(a) = \sum_{b \in B} \mu(a, b)$ and $\pi_2^\sharp(\mu)(b) = \sum_{a \in A} \mu(a, b)$. When $f : A \to \mathbb{D}(B)$, we will abuse notation and write the lifting $f^\sharp : \mathbb{D}(A) \to \mathbb{D}(B)$ to mean $f^\sharp(\mu) \triangleq \mathbb{E}_{x\sim\mu}[f(x)]$.

Finally, if $\alpha : A \to \mathbb{R}^+$, we write $\alpha[X] \in \mathbb{R}^+ \cup \{\infty\}$ for $\sum_{x \in X} \alpha(x)$. Moreover, if $\alpha : A \times B \to \mathbb{R}^+$, we write $\alpha[X, Y]$ (resp. $\alpha[x, Y]$, $\alpha[X, y]$) for $\alpha[X \times Y]$ (resp. $\alpha[\{x\} \times Y$, $\alpha[X \times \{y\}]$). Note that for a sub-distribution $\mu \in \mathbb{D}(A)$ and an event $E \subseteq A$, $\mathbb{P}_\mu[E] = \mu[E]$.

We now review the definition of differential privacy.

▶ **Definition 2** (Dwork et al. [7])**.** A probabilistic computation $M : A \to \mathbb{D}(B)$ satisfies (ϵ, δ)-*differential privacy* w.r.t. an adjacency relation $\phi \subseteq A \times A$ iff for every pair of inputs $a, a' \in A$ such that $a \; \phi \; a'$ and every subset of outputs $E \subseteq B$,

$$\mathbb{P}_{M(a)}[E] \leq e^\epsilon \cdot \mathbb{P}_{M(a')}[E] + \delta.$$

It is useful to define a notion of distance on distributions, reflecting differential privacy.

▶ **Definition 3** (Barthe and Olmedo [5], Barthe et al. [6], Olmedo [10])**.** Let $\epsilon \geq 0$. The ϵ-*DP divergence* $\Delta_\epsilon(\mu_1, \mu_2)$ between two sub-distributions $\mu_1, \mu_2 \in \mathbb{D}(B)$ is defined as

$$\sup_{E \subseteq B} \left(\mathbb{P}_{\mu_1}[E] - e^\epsilon \cdot \mathbb{P}_{\mu_2}[E]\right).$$

Then, differential privacy admits an alternative characterization based on DP divergence.

▶ **Lemma 4.** *A probabilistic computation $M : A \to \mathbb{D}(B)$ satisfies* (ϵ, δ)-differential privacy *w.r.t. an adjacency relation $\phi \subseteq A \times A$ iff $\Delta_\epsilon(M(a), M(a')) \leq \delta$ for every pair of inputs $a, a' \in A$ such that $a \; \phi \; a'$.*

Our new definition of approximate lifting is inspired by a version of approximate liftings involving two witness distributions, proposed by Barthe and Olmedo [6], Olmedo [10].

▶ **Definition 5** (Barthe and Olmedo [6], Olmedo [10]). Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$ be sub-distributions, $\epsilon, \delta \in \mathbb{R}^+$ and $\mathcal{R}$ be a binary relation over A & B. An (ϵ, δ)-*approximate* 2-*lifting* of μ_1 & μ_2 for $\mathcal{R}$ is a pair $(\mu_\triangleleft, \mu_\triangleright)$ of sub-distributions over $A \times B$ s.t.

1. $\pi_1^\sharp(\mu_\triangleleft) = \mu_1$ and $\pi_2^\sharp(\mu_\triangleright) = \mu_2$;
2. $\Delta_\epsilon(\mu_\triangleleft, \mu_\triangleright) \leq \delta$; and
3. $\mathrm{supp}(\mu) \subseteq \mathcal{R}$.

We write $\mu_1 \; \mathcal{R}^{(2)}_{\epsilon,\delta} \; \mu_2$ if there exists an (ϵ, δ)-approximate (2-)lifting of μ_1 & μ_2 for $\mathcal{R}$; the (2) indicates that there are two witnesses in this definition of lifting.

Combined with Lemma 4, a probabilistic computation $M : A \to \mathbb{D}(B)$ is (ϵ, δ)-differentially private if and only if for every two adjacent inputs $a \; \phi \; a'$, there is an approximate lifting of the equality relation: $M(a) =^{(2)}_{\epsilon,\delta} M(a')$.

2-liftings can be generalized by varying the notion of distance given by Δ_ϵ; we will return to this point in Section 6. These liftings also satisfy useful theoretical properties, but some of the properties are not as general as we would like. For example, it is known that 2-liftings satisfy the following mapping property.

▶ **Theorem 6** (Barthe et al. [2]). *Let $\mu_1 \in \mathbb{D}(A_1)$, $\mu_2 \in \mathbb{D}(A_2)$, $f_1 : A_1 \to B_1$, $f_2 : A_2 \to B_2$* surjective *maps and $\mathcal{R}$ a binary relation on B_1 & B_2. Then*

$$f_1^\sharp(\mu_1) \; \mathcal{R}^{(2)}_{\epsilon,\delta} \; f_2^\sharp(\mu_2) \iff \mu_1 \; \mathcal{S}^{(2)}_{\epsilon,\delta} \; \mu_2$$

where $a_1 \; \mathcal{S} \; a_2 \overset{\triangle}{\iff} f_1(a_1) \; \mathcal{R} \; f_2(a_2)$.

This property can be used to pull back an approximate lifting on two distributions over B_1, B_2 to an approximate lifting on two distributions over A_1, A_2. For applications in program logics, B_1, B_2 could be the domain of a program variable, A_1, A_2 could be the set of memories, and f_1, f_2 could project a memory to a program variable. While the mapping theorem is quite useful, it is puzzling why it only applies to surjective maps. For instance, this theorem cannot be used when the maps f_1, f_2 embed a smaller space into a larger space.

For another example, there exist 2-liftings of the following form, sometimes called the *optimal subset coupling*.

▶ **Theorem 7** (Barthe et al. [2]). *Let $\mu \in \mathbb{D}(A)$ and consider two subsets $P_1 \subseteq P_2 \subseteq A$. Suppose that P_2 is a* strict subset *of A. Then, we have the following equivalence:*

$$\mathbb{P}_\mu[P_2] \leq e^\epsilon \cdot \mathbb{P}_\mu[P_1] \iff \mu \; \mathcal{R}^{(2)}_{\epsilon,0} \; \mu,$$

where $a_1 \; \mathcal{R} \; a_2 \overset{\triangle}{\iff} a_1 \in P_1 \iff a_2 \in P_2$.

In this construction, it is puzzling why the larger subset P_2 must be a *strict* subset of the domain A. For example, this theorem does not apply for $P_2 = A$, but we may be able to construct the approximate lifting if we simply embed A into a larger space B – even though μ has support over A! Furthermore, it is not clear why the subsets must be nested, nor is it clear why we can only relate μ to itself.

These shortcomings suggest that the definition of 2-liftings may be problematic. While the distance condition appears to be the most constraining requirement, the marginal and support conditions are responsible for the main issues.

Witnesses can only use pairs in the relation

For some relations $\mathcal{R}$, there may be elements a such that $a\ \mathcal{R}\ b$ does not hold for any b, or vice versa. It can be impossible find witnesses with the correct marginals on these elements, even if the distance condition can be easily satisfied. For instance, we can sometimes construct a pair $\mu_\triangleleft$ and $\mu_\triangleright$ satisfying the distance requirement, but where $\mu_\triangleright$ needs additional mass to achieve the marginal requirement for an element b. Adding this mass anywhere preserves the distance bound, but there may not be an element a such that $a\ \mathcal{R}\ b$.

No canonical choice of witnesses

A related problem is that the marginal requirement only constrains one marginal of each witness distribution. Along the other component, the witnesses may place the mass anywhere on any pair in the relation. As a result, witnesses to an approximate lifting $\mu_1\ \mathcal{R}^{(2)}_{\epsilon,\delta}\ \mu_2$ may have mass outside of $\mathrm{supp}(\mu_1) \times \mathrm{supp}(\mu_2)$, even though it seems that only elements in the support should be relevant to the lifting.

3 $\star$-Liftings and Strassen's Theorem

To improve the theoretical properties of 2-liftings, we propose a simple extension: allow witnesses to be distributions over a larger set.

▶ **Notation 8.** Let A be a set. We write $A^\star$ for $A \uplus \{\star\}$.

▶ **Definition 9** ($\star$-lifting)**.** Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$ be sub-distributions, $\epsilon, \delta \in \mathbb{R}^+$ and $\mathcal{R}$ be a binary relation over A & B. An (ϵ,δ)-*approximate* $\star$-*lifting* of μ_1 & μ_2 for $\mathcal{R}$ is a pair of sub-distributions $\eta_\triangleleft \in \mathbb{D}(A \times B^\star)$ and $\eta_\triangleright \in \mathbb{D}(A^\star \times B)$ s.t.

1. $\pi_1^\sharp(\eta_\triangleleft) = \mu_1$ and $\pi_2^\sharp(\eta_\triangleright) = \mu_2$;
2. $\mathrm{supp}(\eta_{\triangleleft|A\times B}), \mathrm{supp}(\eta_{\triangleright|A\times B}) \subseteq \mathcal{R}$; and
3. $\Delta_\epsilon(\overline{\eta_\triangleleft}, \overline{\eta_\triangleright}) \leq \delta$, where $\overline{\eta_\bullet}$ is the canonical lifting of $\eta_\bullet$ to $A^\star \times B^\star$.

We write $\mu_1\ R^{(\star)}_{\epsilon,\delta}\ \mu_2$ if there exists an (ϵ,δ)-approximate lifting of μ_1 & μ_2 for $\mathcal{R}$.

By adding an element $\star$, we address both problems discussed at the end of the previous section. First, for every $a \in A$, witnesses may place mass at $(a,\star)$; for every $b \in B$, witnesses may place mass at $(\star,b)$. Second, $\star$ can serve as a generic element where all mass that lies outside the supports $\mathrm{supp}(\mu_1) \times \mathrm{supp}(\mu_2)$ may be placed, while preserving the marginal and distance requirements, giving more control over the form of the witnesses.

▶ **Lemma 10.** *Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$ be distributions such that $\mu_1\ \mathcal{R}^{(\star)}_{\epsilon,\delta}\ \mu_2$. Then, there are witnesses with support contained in $\mathrm{supp}(\mu_1)^\star \times \mathrm{supp}(\mu_2)^\star$.*

3.1 Basic Properties

$\star$-liftings satisfy all basic properties satisfied by other notions of lifting. We start by proving that this new definition of lifting still characterizes differential privacy.

▶ **Lemma 11.** *A randomized algorithm $P : A \to \mathbb{D}(B)$ is (ϵ,δ)-differentially private for ϕ iff for all $a_1, a_2 \in A$, $a_1\ \phi\ a_2$ implies $P(a_1) =^{(\star)}_{\epsilon,\delta} P(a_2)$.*

The next lemma establishes several other basic properties of $\star$-liftings: monotonicity, and closure under relational and sequential composition.

▶ **Lemma 12.**

- *Let $\mu_1 \in \mathbb{D}(A)$, $\mu_2 \in \mathbb{D}(B)$, and $\mathcal{R}$ be a binary relation over A & B. If $\mu_1 \ \mathcal{R}^{(\star)}_{\epsilon,\delta} \ \mu_2$, then for any $\epsilon' \geq \epsilon$, $\delta' \geq \delta$ and $\mathcal{S} \supseteq \mathcal{R}$, we have $\mu_1 \ \mathcal{S}^{(\star)}_{\epsilon',\delta'} \ \mu_2$.*
- *Let $\mu_1 \in \mathbb{D}(A)$, $\mu_2 \in \mathbb{D}(B)$, $\mu_2 \in \mathbb{D}(C)$ and $\mathcal{R}$ (resp. $\mathcal{S}$) be a binary relation over A & B (resp. over B & C). If $\mu_1 \ \mathcal{R}^{(\star)}_{\epsilon,\delta} \ \mu_2$ and $\mu_2 \ \mathcal{S}^{(\star)}_{\epsilon',\delta'} \ \mu_3$, then $\mu_1 \ (\mathcal{S} \circ \mathcal{R})^{(\star)}_{\epsilon+\epsilon',\delta+e^{\epsilon}\cdot\delta'} \ \mu_3$.*
- *For $i \in \{1,2\}$, let $\mu_i \in \mathbb{D}(A_i)$ and $\eta_i : A_i \to \mathbb{D}(B_i)$. Let $\mathcal{R}$ (resp. $\mathcal{S}$) be a binary relation over A_1 & A_2 (resp. over B_1 & B_2). If $\mu_1 \ \mathcal{R}^{(\star)}_{\epsilon,\delta} \ \mu_2$ for some $\epsilon, \delta \geq 0$ and for any $(a_1, a_2) \in \mathcal{R}$, $\eta_1(a_1) \ \mathcal{S}^{(\star)}_{\epsilon',\delta'} \ \eta_2(a_2)$ for some $\epsilon', \delta' \geq 0$, then*

  $$\mathbb{E}_{\mu_1}[\eta_1] \ \mathcal{S}^{(\star)}_{\epsilon+\epsilon',\delta+\delta'} \ \mathbb{E}_{\mu_2}[\eta_2].$$

3.2 Equivalence with Sato's Definition

In recent work on verifying differential privacy over general, continuous distributions, Sato [11] proposes an alternative definition of approximate lifting. In the special case of discrete distributions, where measurability of events can be forgotten, his definition can be stated as follows.

▶ **Definition 13** (Sato [11])**.** Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$, $\mathcal{R}$ be a binary relation over A & B and $\epsilon, \delta \geq 0$. Then, there is an (ϵ, δ)-approximate lifting of μ_1 & μ_2 for $\mathcal{R}$ if

$$\forall X \subseteq A.\, \mu_1[X] \leq e^{\epsilon} \cdot \mu_2[\mathcal{R}(X)] + \delta.$$

Notice that this definition has no witness distributions at all; instead, it uses a universal quantifier over all subsets. We can show that ⋆-liftings are equivalent to Sato's definition in the case of discrete distributions. This equivalence is reminiscent of Strassen's theorem from probability theory, which characterizes the existence of probabilistic couplings.

▶ **Theorem 14** (Strassen [12])**.** *Let $\mu_1 \in \mathbb{D}(A)$, $\mu_2 \in \mathbb{D}(B)$ be two proper distributions, and $\mathcal{R}$ let be a binary relation over A & B. Then there exists a joint distribution $\mu \in \mathbb{D}(A \times B)$ with support in $\mathcal{R}$ such that $\pi_1^{\sharp}(\mu) = \mu_1$ and $\pi_2^{\sharp}(\mu) = \mu_2$ if and only if*

$$\forall X \subseteq A.\, \mu_1[X] \leq \mu_2[\mathcal{R}(X)].$$

Our result (Theorem 19) can be viewed as a generalization of Strassen's theorem to approximate couplings. The key ingredient in our proof is the *max-flow min-cut* theorem for *countable* networks; we begin by reviewing the basic setting.

▶ **Definition 15** (Flow network)**.** A *flow network* is a structure $((V, E), \top, \bot, c)$ s.t. $\mathcal{N} = (V, E)$ is a loop-free directed graph without infinite simple path (or rays), $\top$ and $\bot$ are two distinct distinguished vertices of $\mathcal{N}$ s.t. no edge starts from $\bot$ and ends at $\top$, and $c : E \to \mathbb{R}^+ \cup \{+\infty\}$ is a function assigning to each edge of $\mathcal{N}$ a capacity. The capacity c is extended to V^2 by assigning capacity 0 to any pair (u, v) s.t. $(u, v) \notin E$.

▶ **Definition 16** (Flow)**.** Given a flow network $\mathcal{N} \triangleq ((V, E), \top, \bot, c)$, a function $f : V^2 \to \mathbb{R}$ is a *flow* for $\mathcal{N}$ iff

1. $\forall u, v \in V.\, f(u, v) \leq c(u, v)$,
2. $\forall u, v \in V.\, f(u, v) = -f(v, u)$, and
3. $\forall u \in V.\, u \notin \{\top, \bot\} \implies \sum_{v \in V} f(u, v) = 0$ (Kirchhoff's Law).

The *mass* $|f|$ of a flow f is defined as $|f| \triangleq \sum_{v \in V} f(\top, v) \in \mathbb{R}\{\cup + \infty\}$.

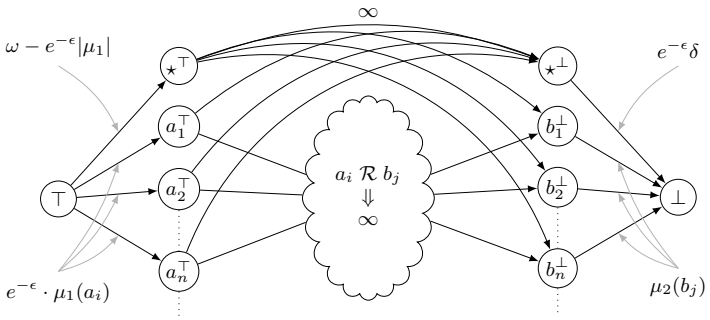


Figure 1 Flow Network in Theorem 19.

▶ **Definition 17** (Cut). Given a flow network $\mathcal{N} \triangleq ((V, E), \top, \perp, c)$, a *cut* for $\mathcal{N}$ is any set $C \subseteq V$ that partition V s.t. $\top \in V$ but $\perp \notin V$. The *cut-set* $\mathcal{E}(C)$ of a cut C is defined as: $\{(u, v) \in E \mid u \in S, v \notin S\}$. The *capacity* $|C| \in \mathbb{R}^+ \cup \{\infty\}$ of a cut is defined as $|C| \triangleq \sum_{(u,v)\in\mathcal{E}(C)} c(u, v)$.

For flow networks with finitely many vertices an edges, the maximum flow is equal to the minimum cut. Aharoni et al. [1] consider when this is the case for a countable network. For the flow networks that we consider in this paper – where there are no infinite directed paths – equality holds.

▶ **Theorem 18** (Weak Countable Max-Flow Min-Cut). *Let $\mathcal{N}$ be a network flow. Then,*

$$\sup\{|f| \mid f \text{ is a flow for } \mathcal{N}\} = \inf\{|C| \mid C \text{ is a cut for } \mathcal{N}\}$$

and both the supremum and infimum are reached.

We are now ready to prove an approximate version of Strassen's theorem, thereby showing equivalence between $\star$-liftings and Sato's liftings.

▶ **Theorem 19.** *Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$, $\mathcal{R}$ be a binary relation over A & B and $\epsilon, \delta \in \mathbb{R}^+$. Then, $\mu_1 \, R^{(\star)}_{\epsilon,\delta} \, \mu_2$ iff $\forall X \subseteq A. \, \mu_1(X) \leq e^\epsilon \cdot \mu_2(\mathcal{R}(X)) + \delta$.*

Proof. We only detail the reverse direction. We can assume that A and B are countable; in the case where A and B are not both countable, we first consider the restriction of μ_1 and μ_2 to their respective supports – which are countable sets – and construct witnesses to the $\star$-lifting. The witnesses can then be extended to a coupling of μ_1 and μ_2 by adding a null mass to the extra points.

Let $\omega \triangleq |\mu_2| + e^{-\epsilon} \cdot \delta$ and let $\top$ and $\perp$ be fresh symbols. For any set X, define $X^\top$ and $X^\perp$ resp. as $\{x^\top \mid x \in X\}$ and $\{x^\perp \mid x \in X\}$. Let $\mathcal{N}$ be the flow network of Figure 1 whose resp. source and sink are $\top$ and $\perp$, whose set of vertices V is $\{\top, \perp\} \uplus (A^\star)^\top \uplus (B^\star)^\perp$, and whose set of edges E is $E_\top \uplus E_\perp \uplus E_\mathcal{R} \uplus E_\star$ with

$$E_\top \triangleq \{\top \mapsto_{\mu_1(a)} a^\top \mid a \in A\} \qquad E_\perp \triangleq \{b^\perp \mapsto_{e^{-\epsilon}\mu_2(b)} \perp \mid b \in B\}$$
$$E_\mathcal{R} \triangleq \{a^\top \mapsto_\infty b^\perp \mid a \, \mathcal{R} \, b \vee a = \star \vee b = \star\} \qquad E_\star \triangleq \{\top \mapsto_{(\omega - e^{-\epsilon}|\mu_1|)} \star^\top, \; \star^\perp \mapsto_{e^{-\epsilon}\delta} \perp\}.$$

Let C be a cut of $\mathcal{N}$ – in the following, we use C independently for the cut C and its cut-set $\mathcal{E}(C)$. We check $|C| \geq \omega$. If $C \cap E_\mathcal{R} \neq \emptyset$ then $|C| = \infty$. Note that $C \cap E_\star = \emptyset$ implies $C \cap E_\mathcal{R} \neq \emptyset$. If $(\top, \star^\top) \in C$ and $(\perp, \star^\perp) \notin C$ then we must have $E_\top \subseteq C$. This implies that $|C| \geq \omega$ since $E_\top \uplus \{(\top, \star^\top)\}$ is a cut with capacity ω. If $(\top, \star^\top) \notin C$ and $(\perp, \star^\perp) \in C$ then

we have $|C| \geq \omega$ in the similar way as above. Otherwise (i.e. $C \cap E_{\mathcal{R}} = \emptyset$ and $E_\star \subseteq C$), for C to be a cut, we must have $\mathcal{R}(A - A^\dagger) \subseteq B^\dagger$ where $A^\dagger \triangleq \{x \in A \mid (\top, x^\top) \in C\}$ and $B^\dagger \triangleq \{y \in B \mid (y^\perp, \perp) \in C\}$. Thus,

$$\begin{aligned}|C| &= e^{-\epsilon} \cdot \mu_1[A^\dagger] + \mu_2[B^\dagger] + |E_\star| \\ &\geq e^{-\epsilon} \cdot \mu_1[A^\dagger] + \mu_2[\mathcal{R}(A - A^\dagger)] + e^{-\epsilon} \cdot \delta + (\omega - e^{-\epsilon} \cdot |\mu_1|) \\ &\geq e^{-\epsilon} \cdot (\mu_1[A^\dagger] + \mu_1[A - A^\dagger]) + \omega - e^{-\epsilon} \cdot |\mu_1| = \omega.\end{aligned}$$

Hence, $E_\top \uplus \{(\star^\perp, \perp)\}$ is a minimum cut with capacity ω. By Theorem 18, we obtain a maximum flow f with mass ω. Note that the flow f saturates the capacity of all edges in $E_\top$, $E_\perp$, and $E_\star$. Let $\hat{f} : (a, b) \in A^\star \times B^\star \mapsto f(a^\top, b^\perp)$. We now define the following distributions:

$$\begin{aligned}\eta_\triangleleft : A \times B^\star &\to \mathbb{R}^+ & \eta_\triangleright : A^\star \times B &\to \mathbb{R}^+ \\ (a, b) &\mapsto e^\epsilon \cdot \hat{f}(a, b) & (a, b) &\mapsto \hat{f}(a, b).\end{aligned}$$

We clearly have $\pi_1^\sharp(\eta_\triangleleft) = \mu_1$ and $\pi_2^\sharp(\eta_\triangleright) = \mu_2$. Moreover, by construction of the flow network $\mathcal{N}$, $\mathrm{supp}(\hat{f}_{|A \times B}) \subseteq \mathcal{R}$. Hence, $\mathrm{supp}(\eta_{\triangleleft|A \times B}), \mathrm{supp}(\eta_{\triangleright|A \times B}) \subseteq \mathcal{R}$. It remains to show that $\Delta_\epsilon(\overline{\eta_\triangleleft}, \overline{\eta_\triangleright}) \leq \delta$. Let X be a subset of $A^\star \times B^\star$. Let $\overline{X_a} \triangleq \{a \in A \mid (a, \star) \in X\}$, $\overline{X_b} \triangleq \{b \in B \mid (\star, b) \in X\}$ and $\overline{X} \triangleq X \cap (A \times B)$. Then,

$$\begin{aligned}\overline{\eta_\triangleleft}[X] - e^\epsilon \cdot \overline{\eta_\triangleright}[X] &= e^\epsilon \left(\hat{f}[\overline{X}] + \hat{f}[\overline{X_a} \times \{\star\}]\right) - e^\epsilon \left(\hat{f}[\overline{X}] + \hat{f}[\{\star\} \times \overline{X_b}]\right) \\ &\leq e^\epsilon \cdot \hat{f}[\overline{X_a} \times \{\star\}] \leq e^\epsilon \cdot \hat{f}[A \times \{\star\}] = \delta.\end{aligned}$$

The last equality holds by Kirchhoff's law: $\hat{f}[A \times \{\star\}] = \sum_{a \in A} f(a^\top, \star^\perp) = f(\star^\perp, \perp) = e^{-\epsilon} \cdot \delta$. ◀

4 Properties of ⋆-Liftings

Our main theorem can be used to show a variety of natural properties of ⋆-liftings. To begin, we can generalize the mapping property from Theorem 6, lifting the requirement that the maps must be surjective.

▶ **Lemma 20.** *Let $\mu_1 \in \mathbb{D}(A_1)$, $\mu_2 \in \mathbb{D}(A_2)$, $f_1 : A_1 \to B_1$, $f_2 : A_2 \to B_2$ and $\mathcal{R}$ a binary relation on B_1 & B_2. Let $\mathcal{S}$ such that $a_1 \mathrel{\mathcal{S}} a_2 \iff f_1(a_1) \mathrel{\mathcal{R}} f_2(a_2)$. Then*

$$f_1^\sharp(\mu_1) \mathrel{\mathcal{R}_{\epsilon,\delta}^{(\star)}} f_2^\sharp(\mu_2) \iff \mu_1 \mathrel{\mathcal{S}_{\epsilon,\delta}^{(\star)}} \mu_2.$$

Similarly, we can generalize the existing rules for up-to-bad reasoning (cf. Barthe et al. [2, Theorem 13]), which restrict the post-condition to be equality. There are two versions: the conditional event is either on the left side, or the right side. Note that the resulting index $\overline{\delta}$ are different in the two cases.

▶ **Lemma 21.** *Let $\mu_1 \in \mathbb{D}(A)$, $\mu_2 \in \mathbb{D}(B)$, $\theta \subseteq A$ and $\mathcal{R} \subseteq A \times B$. Assume that $\mu_1 \, (\theta_\triangleleft \implies \mathcal{R})_{\epsilon,\delta}^{(\star)} \, \mu_2$ for some parameters $\epsilon, \delta \geq 0$. Then, $\mu_1 \, \mathcal{R}_{\epsilon,\overline{\delta}}^{(\star)} \, \mu_2$, where $\overline{\delta} \triangleq \delta + \mu_1[\overline{\theta}]$.*

▶ **Lemma 22.** *Let $\mu_1 \in \mathbb{D}(A)$, $\mu_2 \in \mathbb{D}(B)$, $\theta \subseteq B$ and $\mathcal{R} \subseteq A \times B$. Assume that $\mu_1 \, (\theta_\triangleright \implies \mathcal{R})_{\epsilon,\delta}^{(\star)} \, \mu_2$ for some parameters $\epsilon, \delta \geq 0$. Then, $\mu_1 \, \mathcal{R}_{\epsilon,\overline{\delta}}^{(\star)} \, \mu_2$, where $\overline{\delta} \triangleq \delta + e^\epsilon \cdot \mu_2[\overline{\theta}]$.*

As a consequence, an approximately lifted relation can be conjuncted with a one-sided predicate if the δ parameter is increased. This principle is useful for constructing approximate liftings that express *accuracy* bounds: when $\theta_{a,\triangleleft}$ is an event that happens with high probability, we can assume that $\theta_{a,\triangleleft}$ holds if we increase the δ parameter of the approximate lifting.

▶ **Lemma 23.** *Let $\mu_1 \in \mathbb{D}(A)$, $\mu_2 \in \mathbb{D}(B)$, $\theta_a \subseteq A$, $\theta_b \subseteq B$ and $\mathcal{R} \subseteq A \times B$. Assume that $\mu_1 \ \mathcal{R}^{(\star)}_{\epsilon,\delta} \ \mu_2$. Then, $\mu_1 \ (\theta_{a,\triangleleft} \wedge \mathcal{R})^{(\star)}_{\epsilon,\delta_a} \ \mu_2$ and $\mu_1 \ (\theta_{b,\triangleright} \wedge \mathcal{R})^{(\star)}_{\epsilon,\delta_b} \ \mu_2$ where $\delta_a \triangleq \delta + \mu_1[\overline{\theta_a}]$ and $\delta_b \triangleq \delta + e^{\epsilon} \cdot \mu_2[\overline{\theta_b}]$.*

$\star$-liftings also support a significant generalization of optimal subset coupling. Unlike the known construction for 2-liftings (Theorem 7), the two subsets need not be nested, and either subset may be the entire domain. Furthermore, the distributions μ_1, μ_2 need not be the same, or even have the same domain. Finally, the equivalence is valid for any parameters (ϵ, δ), not just $\delta = 0$.

▶ **Theorem 24** (Barthe et al. [2])**.** *Let $\mu_1 \in \mathbb{D}(A_1)$, $\mu_2 \in \mathbb{D}(A_2)$ and consider two subsets $P_1 \subseteq A_1, P_2 \subseteq A_2$. Then, we have the following equivalence:*

$$\mathbb{P}_{\mu_1}[P_1] \leq e^{\epsilon} \cdot \mathbb{P}_{\mu_2}[P_2] + \delta \wedge \mathbb{P}_{\mu_1}[A_1 - P_1] \leq e^{\epsilon} \cdot \mathbb{P}_{\mu_2}[A_2 - P_2] + \delta \iff \mu_1 \ \mathcal{R}^{(\star)}_{\epsilon,\delta} \ \mu_2,$$

where $a_1 \ \mathcal{R} \ a_2 \overset{\triangle}{\iff} a_1 \in P_1 \iff a_2 \in P_2$.

Proof. Immediate by Theorem 19. ◀

Finally, we can directly extend known composition theorems from differential privacy to $\star$-liftings. This connection is quite useful for lifting existing results from the privacy literature–which can be quite sophisticated – to approximate liftings.

▶ **Lemma 25.** *Pose $\mathbb{R}_2^+ \triangleq \mathbb{R}^+ \times \mathbb{R}^+$ and let $(\mathbb{R}_2^+)^*$ be the set of finite sequences over $\mathbb{R}_2^+$. Let $r : (\mathbb{R}_2^+)^* \to \mathbb{R}_2^+$ be a* DP-composition operator*, i.e. r is an operator such that for any sets A, D and family $\{f_i : D \times A \to \mathbb{D}(A)\}_{i<n}$ of functions, if for every $a \in A$ and $i < n$, $f_i(-, a) : D \to \mathbb{D}(A)$ is (ϵ_i, δ_i)-differentially private for some parameters $\epsilon_i, \delta_i \geq 0$ and fixed adjacency relation ϕ, then, for any $a \in A$, $F(-, a)$ is (ϵ^*, δ^*)-differentially private for ϕ, where $F : (d, a) \mapsto (\bigcirc_{i<n} \ (f_i(d, -))^{\sharp})(\mathbb{1}_a)$ is the the n-fold composition of the $[f_i]_{i<n}$ and $(\epsilon^*, \delta^*) \triangleq r([(\epsilon_i, \delta_i)]_{i<n})$.*

Let $n \in \mathbb{N}$ and assume given two families of sets $\{A_i\}_{i\leq n}$ and $\{B_i\}_{i\leq n}$, together with a family of binary relations $\{\mathcal{R}(i) \subseteq A_i \times B_i\}_{i\leq n}$. Fix two families of functions $\{g_i : A_i \to \mathbb{D}(A_{i+1})\}_{i<n}$ and $\{h_i : B_i \to \mathbb{D}(B_{i+1})\}_{i<n}$ s.t. for any $i < n$ and $(a, b) \in \mathcal{R}(i)$ we have:
1. *$g_i(a) \ \mathcal{R}(i+1)^{(\star)}_{\epsilon_i,\delta_i} \ h_i(b)$ for some parameters $\epsilon_i, \delta_i \geq 0$, and*
2. *$g_i(a)$ and $h_i(b)$ are proper distributions.*
Then, for $(a_0, b_0) \in \mathcal{R}_0$, there exists a $\star$-lifting

$$G(a_0) \ \mathcal{R}(n)^{(\star)}_{\epsilon^*,\delta^*} \ H(b_0)$$

where $(\epsilon^*, \delta^*) \triangleq r([(\epsilon_i, \delta_i)]_{i<n})$, and $G : A_0 \to \mathbb{D}(A_n)$ and $H : B_0 \to \mathbb{D}(B_n)$ are the n-fold compositions of $[g_i]_{i\leq n}$ and $[h_i]_{i\leq n}$ respectively – i.e. $G(a) \triangleq (\bigcirc_{i<n} \ g_i^{\sharp})(\mathbb{1}_a)$ and $H(b) \triangleq (\bigcirc_{i<n} \ h_i^{\sharp})(\mathbb{1}_b)$.

For some of the more sophisticated composition results (notably, the advanced composition theorem by Dwork et al. [8]), Lemma 25 is not quite strong enough and requires a slight adaptation of the notion of $\star$-lifting. We refer to the full version of the paper for more details.

5 Comparison with Existing Approximate Liftings

Now that we have seen ⋆-liftings, we briefly consider other definitions of approximate liftings. We have already seen 2-liftings, which involve two witnesses (Definition 5). Evidently, ⋆-liftings strictly generalize 2-liftings.

▶ **Theorem 26.** *For all binary relations $\mathcal{R}$ over A & B and parameters $\epsilon, \delta \geq 0$, we have $\mathcal{R}^{(2)}_{\epsilon,\delta} \subseteq \mathcal{R}^{(\star)}_{\epsilon,\delta}$. There exist relations and parameters where the inclusion is strict.*

Proof. The inclusion $\mathcal{R}^{(2)}_{\epsilon,\delta} \subseteq \mathcal{R}^{(\star)}_{\epsilon,\delta}$ is immediate. We have a strict inclusion $\mathcal{R}^{(2)}_{\epsilon,\delta} \subsetneq \mathcal{R}^{(\star)}_{\epsilon,\delta}$ even for $\delta = 0$ by considering the optimal subset coupling from Theorem 7. Consider a distribution μ over set A, and let $P_1 \subseteq P_2 = A$. There is an $(\epsilon, 0)$-approximate ⋆-lifting (by Theorem 24), but a $(\epsilon, 0)$-approximate 2-lifting does not exist if μ has non-zero mass outside of P_1: the first witness $\mu_\triangleleft$ must place non-zero mass at (a_1, a_2) with $a_1 \notin P_1$ in order to have $\pi_1^\sharp(\mu_\triangleleft) = \mu$, but we must have $a_2 \notin P_2$ for the support requirement, and there is no such a_2. ◀

It is more interesting to compare ⋆-liftings with the original definitions of (ϵ, δ)-approximate lifting, by Barthe et al. [5]. They introduce two notions, a symmetric lifting and an asymmetric lifting, each using a single witness distribution. We will focus on the asymmetric version.

▶ **Definition 27** (Barthe et al. [5])**.** Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$ be sub-distributions, $\epsilon, \delta \in \mathbb{R}^+$ and $\mathcal{R}$ be a binary relation over A & B. An (ϵ, δ)-*approximate* 1-*lifting* of μ_1 & μ_2 for $\mathcal{R}$ is a sub-distribution $\mu \in \mathbb{D}(A \times B)$ s.t.

1. $\pi_1^\sharp(\mu) \leq \mu_1$ and $\pi_2^\sharp(\mu) \leq \mu_2$;
2. $\Delta_\epsilon(\mu_1, \pi_1^\sharp(\mu)) \leq \delta$; and
3. $\mathrm{supp}(\mu) \subseteq \mathcal{R}$.

In the first point we take the point-wise order on sub-distributions: if μ and μ' are sub-distributions over X, then $\mu \leq \mu'$ when $\mu(x) \leq \mu'(x)$ for all $x \in X$. We will write $\mu_1 \; \mathcal{R}^{(1)}_{\epsilon,\delta} \; \mu_2$ if there exists an (ϵ, δ)-approximate 1-lifting of μ_1 & μ_2 for $\mathcal{R}$; the (1) indicates that there is one witness for this lifting.

1-liftings bear a close resemblance to *probabilistic couplings* from probability theory, which also have a single witness. However, 1-liftings are less well-understood theoretically than 2-liftings – basic properties such as mapping (Theorem 20) are not known to hold; the subset coupling (Theorem 7) is not known to exist.

Somewhat surprisingly, 1-liftings are equivalent to ⋆-liftings (and hence by Theorem 19, also to Sato's approximate lifting).

▶ **Theorem 28.** *For all binary relations $\mathcal{R}$ over A & B and parameters $\epsilon, \delta \geq 0$, we have $\mathcal{R}^{(1)}_{\epsilon,\delta} = \mathcal{R}^{(\star)}_{\epsilon,\delta}$.*

6 ⋆-Lifting for f-Divergences

The definition of ⋆-lifting can be extended to lifting constructions based on general f-divergences, as previously proposed by Barthe and Olmedo [6], Olmedo [10]. Roughly, a f-divergence a function $\Delta_f(\mu_1, \mu_2)$ that measures the difference between two probability distributions μ_1 and μ_2. Much like we generalized their definition for (ϵ, δ)-liftings, we can define ⋆-lifting with f-divergences. Before going any further, let us first define formally f-divergences. We denote by $\mathcal{F}$ the set of non-negative convex functions vanishing at 1: $\mathcal{F} = \{f : \mathbb{R}^+ \to \mathbb{R}^+ \mid f(1) = 0\}$. We also adopt the following notational conventions: $0 \cdot f(0/0) \triangleq 0$, and $0 \cdot f(x/0) \triangleq x \cdot \lim_{t \to 0^+} t \cdot f(1/t)$; we write L_f for the limit.

▶ **Definition 29.** Given $f \in \mathcal{F}$, the *f-divergence* $\Delta_f(\mu_1, \mu_2)$ between two distributions μ_1 and μ_2 in $\mathbb{D}(A)$ is defined as:

$$\Delta_f(\mu_1, \mu_1) = \sum_{a \in A} \nu(a) f\left(\frac{\mu_1(a)}{\mu_2(a)}\right).$$

Examples of f-divergences include statistical distance ($f(t) = \frac{1}{2}|t-1|$), Kullback-Leibler divergence ($f(t) = \ln(t) - t + 1$), and Hellinger distance ($f(t) = \frac{1}{2}(\sqrt{t}-1)^2$).

▶ **Definition 30** ($\star$-lifting for f-divergences)**.** Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$ be distributions, $\mathcal{R}$ be a binary relation over A & B, and $f \in \mathcal{F}$. An $(f;\delta)$-*approximate lifting* of μ_1 & μ_2 for $\mathcal{R}$ is a pair of distributions $\eta_\triangleleft \in \mathbb{D}(A \times B^\star)$ and $\eta_\triangleright \in \mathbb{D}(A^\star \times B)$ s.t.

- $\pi_1^\sharp(\eta_\triangleleft) = \mu_1$ and $\pi_2^\sharp(\eta_\triangleright) = \mu_2$;
- $\mathrm{supp}(\eta_{\triangleleft|A\times B}), \mathrm{supp}(\eta_{\triangleright|A\times B}) \subseteq \mathcal{R}$; and
- $\Delta_f(\overline{\eta_\triangleleft}, \overline{\eta_\triangleright}) \leq \delta$,

where $\overline{\eta_\bullet}$ is the canonical lifting of $\eta_\bullet$ to $A^\star \times B^\star$. We will write: $\mu_1 \; R_{f;\delta}^{(\star)} \; \mu_2$ if there exists an $(f;\delta)$-approximate lifting of μ_1 & μ_2 for $\mathcal{R}$.

$\star$-liftings for f-divergences compose sequentially.

▶ **Lemma 31.** *Suppose f has divergence statistical distance, Kullback-Leibler, or Hellinger distance. For $i \in \{1,2\}$, let $\mu_i \in \mathbb{D}(A_i)$ and $\eta_i : A_i \to \mathbb{D}(B_i)$. Let $\mathcal{R}$ (resp. $\mathcal{S}$) be a binary relation over A_1 & A_2 (resp. over B_1 & B_2). If $\mu_1 \; \mathcal{R}_{f;\delta}^{(\star)} \; \mu_2$ for some $\delta \geq 0$ and for any $(a_1, a_2) \in \mathcal{R}$ we have $\eta_1(a_1) \; \mathcal{S}_{f;\delta'}^{(\star)} \; \eta_2(a_2)$ for some $\delta' \geq 0$, then*

$$\mathbb{E}_{\mu_1}[\eta_1] \; \mathcal{S}_{f;\delta+\delta'}^{(\star)} \; \mathbb{E}_{\mu_2}[\eta_2].$$

Much like the $\star$-liftings we saw before, $\star$-liftings for f-divergences have witness distributions with support determined by the support of μ_1 and μ_2 (cf. Lemma 10).

▶ **Lemma 32.** *Let $\mu_1 \in \mathbb{D}(A)$ and $\mu_2 \in \mathbb{D}(B)$ be distributions such that $\mu_1 \; R_{f;\delta}^{(\star)} \; \mu_2$. Then, there are witnesses with support contained in* $\mathrm{supp}(\mu_1)^\star \times \mathrm{supp}(\mu_2)^\star$.

Finally, the mapping property from Lemma 20 holds also for these $\star$-liftings. While the proof of Lemma 20 relies on the equivalence for Sato's definition, there is no such equivalence (or definition) for general f-divergences. Therefore, we must work directly with the witnesses of the approximate lifting.

▶ **Lemma 33.** *Let $\mu_1 \in \mathbb{D}(A_1)$, $\mu_2 \in \mathbb{D}(A_2)$, $g_1 : A_1 \to B_1$, $g_2 : A_2 \to B_2$ and $\mathcal{R}$ a binary relation on B_1 & B_2. Let $\mathcal{S}$ such that $a_1 \; \mathcal{S} \; a_2 \overset{\triangle}{\iff} g_1(a_1) \; \mathcal{R} \; g_2(a_2)$. Then*

$$g_1^\sharp(\mu_1) \; \mathcal{R}_{f;\delta}^{(\star)} \; g_2^\sharp(\mu_2) \iff \mu_1 \; \mathcal{S}_{f;\delta}^{(\star)} \; \mu_2.$$

7 Conclusion

We have proposed a new definition of approximate lifting that unifies existing constructions and satisfies an approximate variant of Strassen's theorem. Our notion is useful both to simplify the soundness proof of existing program logics and to strengthen some of their proof rules. We see at least two important directions for future work. First, adapting existing program logics (for instance, apRHL [5]) to use $\star$-liftings, and formalizing examples that were out of reach of previous systems. Second, our notion of $\star$-liftings only applies when distributions have discrete support. It would be interesting to see if $\star$-liftings – and the approximate Strassen's theorem – can be generalized to the continuous setting.

Acknowledgments. We thank the anonymous reviewers for their helpful suggestions.

References

1 Ron Aharoni, Eli Berger, Agelos Georgakopoulos, Amitai Perlstein, and Philipp Sprüssel. The max-flow min-cut theorem for countable networks. *J. Comb. Theory, Ser. B*, 101(1):1–17, 2011. doi:10.1016/j.jctb.2010.08.002.

2 Gilles Barthe, Noémie Fong, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Advanced probabilistic couplings for differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS), Vienna, Austria*, 2016. URL: https://arxiv.org/abs/1606.07143.

3 Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *IEEE Symposium on Logic in Computer Science (LICS), New York, New York*, 2016. URL: http://arxiv.org/abs/1601.05047.

4 Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin C. Pierce. Programming language techniques for differential privacy. *SIGLOG News*, 3(1):34–53, 2016. doi:10.1145/2893582.2893591.

5 Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella-Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Transactions on Programming Languages and Systems*, 35(3):9, 2013. URL: http://software.imdea.org/~bkoepf/papers/toplas13.pdf.

6 Gilles Barthe and Federico Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. In *International Colloquium on Automata, Languages and Programming (ICALP), Riga, Latvia*, volume 7966 of *Lecture Notes in Computer Science*, pages 49–60. Springer-Verlag, 2013. URL: http://certicrypt.gforge.inria.fr/2013.ICALP.pdf.

7 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *IACR Theory of Cryptography Conference (TCC), New York, New York*, pages 265–284, 2006. doi:10.1007/11681878_14.

8 Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *IEEE Symposium on Foundations of Computer Science (FOCS), Las Vegas, Nevada*, pages 51–60, 2010. URL: http://research.microsoft.com/pubs/155170/dworkrv10.pdf.

9 Torgny Lindvall. *Lectures on the coupling method*. Courier Corporation, 2002.

10 Federico Olmedo. *Approximate Relational Reasoning for Probabilistic Programs*. PhD thesis, Universidad Politécnica de Madrid, 2014. URL: http://software.imdea.org/~federico/thesis.pdf.

11 Tetsuya Sato. Approximate relational Hoare logic for continuous random samplings. In *Conference on the Mathematical Foundations of Programming Semantics (MFPS), Pittsburgh, Pennsylvania*, volume 325 of *Electronic Notes in Theoretical Computer Science*, pages 277–298. Elsevier, 2016. URL: https://arxiv.org/abs/1603.01445.

12 Volker Strassen. The existence of probability measures with given marginals. *The Annals of Mathematical Statistics*, pages 423–439, 1965. URL: http://projecteuclid.org/euclid.aoms/1177700153.

13 Hermann Thorisson. *Coupling, Stationarity, and Regeneration*. Springer-Verlag, 2000.

14 Cédric Villani. *Optimal transport: old and new*. Springer-Verlag, 2008.

Bisimulation Metrics for Weighted Automata*†

Borja Balle[1], Pascale Gourdeau[2], and Prakash Panangaden[3]

1 **Department of Mathematics and Statistics, Lancaster University, Lancaster, UK**
b.deballepigem@lancaster.ac.uk
2 **School of Computer Science, McGill University, Montreal, Quebec, Canada**
pascale.gourdeau@mail.mcgill.ca
3 **School of Computer Science, McGill University, Montreal, Quebec, Canada**
prakash@cs.mcgill.ca

Abstract

We develop a new bisimulation (pseudo)metric for weighted finite automata (WFA) that generalizes Boreale's linear bisimulation relation. Our metrics are induced by seminorms on the state space of WFA. Our development is based on spectral properties of sets of linear operators. In particular, the joint spectral radius of the transition matrices of WFA plays a central role. We also study continuity properties of the bisimulation pseudometric, establish an undecidability result for computing the metric, and give a preliminary account of applications to spectral learning of weighted automata.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Weighted automata, bisimulation, metrics, spectral theory, learning

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.103

1 Introduction

Weighted finite automata (WFA) form a fundamental computational model that subsumes probabilistic automata and various other types of quantitative automata. They are much used in machine learning and natural language processing, and are certainly relevant to quantitative verification and to the theory of control systems [16]. The theory of minimization of weighted finite automata goes back to Schützenberger [30] which implicitly exploits duality as made explicit in [9]. In [6] we began studying *approximate* minimization of WFA by using spectral methods. The idea there was to obtain automata for a given weighted language, smaller than the minimal possible which, of course, means that the automaton constructed does not *exactly* recognize the given weighted language but comes "close enough."

In [6] the notion of proximity to the desired language was captured by an ℓ_2 distance. However, a powerful technique for understanding approximate behavioural equivalence is by using more general *behavioural metrics*. In particular, with a behavioural pseudometric we recover bisimulation as the kernel. Such behavioural metrics for Markov processes were proposed by Giacalone et al. [20] and the first successful pseudometric that has bisimulation as its kernel is due to Desharnais et al. [14, 15]; see [28] for an expository account. The subject was greatly developed by van Breugel and Worrell [32] among others. For WFA, a beautiful treatment of linear bisimulation relations was given by Boreale [10]. We were

* The full version of this paper, including the appendix, can be found at https://arxiv.org/abs/1702.08017.

† This research has been supported by a grant from NSERC (Canada).

motivated to develop a metric analogue of Boreale's linear bisimulation with the eventual goal of using it to analyze approximate minimization. In the present paper we develop the general theory of bisimulation (pseudo)metrics for WFA (and for weighted languages) deferring the application to approximate minimization to future work.

It turns out that in the linear algebraic setting appropriate to WFA it is a (semi)norm rather than a (pseudo)metric that is the fundamental quantity of interest. Indeed, as one might expect, in a vector space setting norms and seminorms are the natural objects from which metrics and pseudometrics can be derived. The bisimulation metric that we construct actually comes from a bisimulation seminorm which is obtained, as usual, using the Banach fixed-point theorem. Interestingly, we also provide a closed-form expression for the fixed point bisimulation seminorm and use it to study several of its properties.

Our main contributions are:

1. The construction of bisimulation seminorms and the associated pseudometric on WFA (Section 3). The existence of the fixed point depends on some delicate applications of spectral theory, specifically the joint spectral radius of a set of matrices.
2. We obtain metrics on the space of weighted languages from the metrics on WFA (Section 3).
3. We show two continuity properties of the metric; one using definitions due to Jaeger et al. [24] and the other developed here (Section 4).
4. We show undecidability results for computing our metrics (Section 5).
5. Nevertheless, we show that one can successfully exploit these metrics for applications in machine learning (Section 6).

The metric of the present paper led naturally to some sophisticated topological and spectral theory arguments which one would not have anticipated from the treatment of linear bisimulation in [10].

2 Background

In this section we recall preliminary definitions and results that will be used throughout the rest of the paper. We assume the reader is familiar with norms and vector spaces; these topics are reviewed in the appendix [4]. Here we discuss Boreale's linear bisimulation relations for weighted automata and provide a short primer on the joint spectral radius of a set of linear operators.

2.1 Strings and Weighted Automata

Given a finite alphabet Σ we let $\Sigma^\star$ denote the set of all finite strings with symbols in Σ and let Σ^∞ denote the set of all infinite strings with symbols in Σ and we write $\Sigma^\omega = \Sigma^\star \cup \Sigma^\infty$. The length of a string $x \in \Sigma^\omega$ is denoted by $|x|$; $|x| = \infty$ whenever $x \in \Sigma^\infty$. Given a string $x \in \Sigma^\omega$ and an integer $0 \leq t \leq |x|$ we write $x_{\leq t}$ to denote the prefix containing the first t symbols from x, with $x_{\leq 0} = \epsilon$. Given an integer $t \geq 0$ we will write Σ^t (resp. $\Sigma^{\leq t}$) for the set of all strings with length equal to (resp. at most) t. The reverse of a finite string $x = x_1 x_2 \cdots x_t$ is given by $\bar{x} = x_t x_{t-1} \cdots x_1$.

We only consider automata with weights in the real field $\mathbb{R}$. We will mostly be concerned with properties of weighted automata that are invariant under change of basis. Accordingly, our presentation uses weighted automata whose state space is an abstract real vector space.

A weighted finite automaton (WFA) is a tuple $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma \in \Sigma} \rangle$ where Σ is a finite alphabet, V is a finite-dimensional vector space, $\alpha \in V$ is a vector representing the initial weights, $\beta \in V^*$ is a linear form representing the final weights, and $\tau_\sigma : V \to V$ is a

linear map representing the transition indexed by $\sigma \in \Sigma$. The vectors in V are called states of A. We shall denote by $n = \dim(A) = \dim(V)$ the dimension of A. The transition maps τ_σ can be extended to arbitrary finite strings in the obvious way.

A weighted automaton $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ computes the function $f_A : \Sigma^\star \to \mathbb{R}$ (sometimes also referred to as the weighted language in $\mathbb{R}^{\Sigma^\star}$ recognized by A) given by $f_A(x) = \beta(\tau_x(\alpha))$. Given a WFA $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ and a state $v \in V$ we define the weighted automaton $A_v = \langle \Sigma, V, v, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ obtained from A by taking v as the initial state. We call f_{A_v} the function realized by state v. Similarly, give a linear form $w \in V^*$ we define the weighted automaton $A^w = \langle \Sigma, V, \alpha, w, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ where the final weights are replaced by w. The reverse of a weighted automaton A is $\bar{A} = \langle \Sigma, V^*, \beta, \alpha, \{\tau_\sigma^\top\}_{\sigma\in\Sigma}\rangle$, where $\tau_\sigma^\top : V^* \to V^*$ is the transpose of τ_σ. It is easy to check that the function computed by $\bar{A}$ satisfies $f_{\bar{A}}(x) = f_A(\bar{x})$ for all $x \in \Sigma^\star$.

2.2 Linear Bisimulations

Linear bisimulations for weighted automata were introduced by Boreale in [10]. Here we recall the key definition and several important facts.

▶ **Definition 1.** A *linear bisimulation* for a weighted automaton $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ on a vector space V is a linear subspace $W \subseteq V$ satisfying the following two conditions:

1. $\beta(v) = 0$ for all $v \in W$; that is, $W \subseteq \ker(\beta)$, and
2. W is invariant by each τ_σ; that is, $\tau_\sigma(W) \subseteq W$ for all $\sigma \in \Sigma$.

Furthermore, two states $u, v \in V$ are called W-*bisimilar* if $u - v \in W$.

In particular, the trivial subspace $W = \{0\}$ is always a linear bisimulation. The notion of W-bisimilarity induces an equivalence relation on V which we will denote by $\sim_W$. The kernel of an equivalence relation $\sim$ on a vector space V is the set of vectors in the equivalence class of the null vector: $\ker(\sim) = \{v \in V : v \sim 0\}$. It is immediate from the definition that for any bisimulation relation $\sim_W$ we have $\ker(\sim_W) = W$.

Given a weighted automaton A we say that $u, v \in V$ are A-bisimilar if there exists a bisimulation W for A such that $u \sim_W v$. The corresponding equivalence relation is denoted by $\sim_A$. Boreale showed in [10] that for every WFA A there exists a bisimulation W_A such that $\sim_{W_A}$ exactly coincides with $\sim_A$, and the bisimulation can be obtained as $W_A = \ker(\sim_A)$. He also showed that W_A is in fact the largest linear bisimulation for A in the sense that any other linear bisimulation W for A must be a subspace of W_A. Accordingly, we shall refer to the relation $\sim_A$ and the subspace W_A as A-bisimulation.

Note that the subspaces considered in Definition 1 are independent of the initial state α of A. In fact, A-bisimilarity can be understood as a relation between possible initial states for A. Indeed, using the definition of $\sim_A$ it is immediate to check that for any states $u, v \in V$ we have $u \sim_A v$ if and only if $f_{A_u} = f_{A_v}$. This implies that in a WFA where the bisimulation W_A corresponding to $\sim_A$ satisfies $W_A = \{0\}$ every state realizes a different function. Such an automaton is called observable. A weighted automaton is called reachable if the reverse $\bar{A}$ is observable.

A weighted automaton A is minimal if for any other weighted automaton A' over the same alphabet such that $f_A = f_{A'}$ we have $\dim(A) \leq \dim(A')$. It is also shown in [10] that linear bisimulations can be used to characterize minimality, in the sense that A is minimal if and only if it is observable and reachable.

2.3 Joint Spectral Radius

The joint spectral radius of a set of linear operators is a natural generalization of the spectral radius of a single linear operator. The joint spectral radius and several equivalent notions have been thoroughly studied since the 1960's. These radiuses arise in many fundamental problems in operator theory, control theory, and computational complexity. See [25] for an introduction to their properties and applications. Here we recall the basic definitions and some important facts related to quasi-extremal norms.

▶ **Definition 2.** The *joint spectral radius* of a collection $M = \{\tau_i\}_{i\in I}$ of linear maps $\tau_i : V \to V$ on a normed vector space $(V, \|\cdot\|)$ is defined as

$$\rho(M) = \limsup_{t\to\infty} \left(\sup_{T\in I^t} \left\| \prod_{i\in T} \tau_i \right\| \right)^{1/t} = \lim_{t\to\infty} \left(\sup_{T\in I^t} \left\| \prod_{i\in T} \tau_i \right\| \right)^{1/t} .$$

The second equality above is a generalization of Gelfand's formula for the spectral radius of a single operator due to Daubechies and Lagarias [11, 12]. An important fact about the joint spectral radius is that $\rho(M)$ is independent of the norm $\|\cdot\|$, i.e. one obtains the same radius regardless of the norm given to the vector space V. The joint spectral radius behaves nicely with respect to direct sums, in the sense that given two sets of operators $M = \{\tau_i\}_{i\in I}$ and $M' = \{\tau'_i\}_{i\in I}$, then $\rho(\{\tau_i \oplus \tau'_i\}_{i\in I}) = \max\{\rho(M), \rho(M')\}$.

The notion of joint spectral radius can be readily extended to weighted automata. Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ be a weighted automaton with states on a normed vector space $(V, \|\cdot\|)$. Then the spectral radius of A is defined as $\rho(A) = \rho(M)$ where $M = \{\tau_\sigma\}_{\sigma\in\Sigma}$. In this case the definition above can be rewritten as

$$\rho(A) = \lim_{t\to\infty} \left(\sup_{x\in\Sigma^t} \|\tau_x\| \right)^{1/t} .$$

Now we discuss several fundamental properties of the joint spectral radius that will play a role in the rest of the paper. Like in the case of the classic spectral radius, the joint spectral radius is upper bounded by the norms of the operators in M: $\rho(M) \leq \sup_{i\in I} \|\tau_i\|$. Obtaining lower bounds for $\rho(M)$ is a major problem directly related to the hardness of computing approximations to $\rho(M)$. An approach often considered in the literature is to search for extremal norms. A norm $\|\cdot\|$ on V is extremal for M if the corresponding induced norm satisfies $\|\tau_i\| \leq \rho(M)$ for all $i \in I$. This immediately implies that given an extremal norm for M we have $\rho(M) = \sup_{i\in I} \|\tau_i\|$. Conditions on M guaranteeing the existence of an extremal norm have been derived by Barabanov and others; see [33] and references therein. However, most of these conditions are quite technical and algorithmically hard to verify. On the other hand, if one only insists on approximate extremality, the following result, due to Rota and Strang, guarantees the existence of such norms for any set of matrices M that is compact with respect to the topology generated by the operator norm in V.

▶ **Theorem 3** ([29])**.** *Let $M = \{\tau_i\}_{i\in I}$ be a compact set of linear maps on V. For any $\eta > 0$ there exists a norm $\|\cdot\|$ on V that satisfies $\|\tau_i(v)\| \leq (\rho(M) + \eta)\|v\|$ for every $i \in I$ and every $v \in V$.*

The statement above is in fact a special case of Proposition 1 in [29]; a proof for finite sets M can be found in [8]. An important result due to Barabanov [7] states that the function $M \mapsto \rho(M)$ defined on compact sets of operators is continuous (see also [22]). Another result that we will need was again proved by Barbanov in [7] and it states that if M is a bounded

set of linear operators and $\bar{M}$ denotes its closure then $\rho(M) = \rho(\bar{M})$. Note that if M is bounded then its closure $\bar{M}$ is compact by the Heine–Borel theorem.

A special case which makes the joint spectral radius easier to work with is when the set of matrices M is irreducible. A set of linear maps M is called *irreducible* if the only subspaces $W \subseteq V$ such that $\tau_i(W) \subseteq W$ for all $i \in I$ are $W = \{0\}$ and $W = V$. If there exists a non-trivial subspace $W \subset V$ invariant by all τ_i we say that M is reducible. In fact, almost all sets of matrices are irreducible in following sense. The Hausdorff distance between two sets of linear maps M and M' on the same normed vector space $(V, \|\cdot\|)$ is given by

$$d_H(M, M') = \max\left\{ \sup_{\tau \in M} \inf_{\tau' \in M'} \|\tau - \tau'\|, \sup_{\tau' \in M'} \inf_{\tau \in M} \|\tau - \tau'\| \right\} .$$

It is possible to show that irreducible sets of matrices are dense among compact sets of matrices with respect to the topology induced by the Haussdorff distance. Furthermore, Wirth showed in [33] that the joint spectral radius is locally Lipschitz continuous around irreducible sets of matrices with respect to the Hausdorff topology (see also [26] for explicit expressions for the Lipschitz constants). This can be seen as an extension of Barabanov's continuity result providing extra information about the behaviour of the function $M \mapsto \rho(M)$.

Again, the concept of irreducibility can be readily extended to WFA. We say that the weighted automaton $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma \in \Sigma} \rangle$ is irreducible if $M = \{\tau_\sigma\}_{\sigma \in \Sigma}$ is irreducible. This concept will play a role in Section 6. The following result provides a characterization of irreducibility for weighted automata in terms of minimality. In particular, the result shows that irreducibility is a stronger condition than minimality. A proof is provided in the appendix [4].

▶ **Theorem 4.** *A weighted automaton $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma \in \Sigma} \rangle$ is irreducible if and only if A_v^w is minimal for all $v \in V$ and $w \in V^*$ with $v \neq 0$ and $w \neq 0$.*

3 Bisimulation Seminorms and Pseudometrics for WFA

In the same way that the largest bisimulation relation in many settings can be obtained as a fixed point of a certain operator on equivalence relations, a possible way to define bisimulation (pseudo)metrics is via a similar fixed-point construction. See [17] for an example in the case of Markov decision processes. In this section, the fixed-point construction is used to obtain a bisimulation seminorm on states of a given WFA. Given two WFA we can build their difference automaton A and compute the corresponding seminorm of the initial state of A. This construction yields a bisimulation pseudometric between weighted automata.

Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma \in \Sigma} \rangle$ be a weighted automaton over the vector space V. Let $\mathcal{S}$ denote the set of all seminorms on V. Given $\gamma > 0$ we define the map $F_{A,\gamma} : \mathcal{S} \to \mathcal{S}$ between seminorms given by

$$F_{A,\gamma}(s)(v) = |\beta(v)| + \gamma \max_{\sigma \in \Sigma} s(\tau_\sigma(v)) \ . \tag{1}$$

Note that this definition is independent of the initial state α, as is the linear bisimulation for A described in Section 2.2. In the sequel we shall write F instead of $F_{A,\gamma}$ whenever A and γ are clear from the context.

To verify that $F : \mathcal{S} \to \mathcal{S}$ is well defined we must check that the image $F(s)$ of any seminorm s is also a seminorm. Absolute homogeneity is immediate by the linearity of β

and τ_σ and the absolute homogeneity of s. For the subadditivity we have

$$\begin{aligned} F(s)(u+v) &= |\beta(u+v)| + \gamma \max_{\sigma\in\Sigma} s(\tau_\sigma(u+v)) \\ &= |\beta(u)+\beta(v)| + \gamma \max_{\sigma\in\Sigma} s(\tau_\sigma(u)+\tau_\sigma(v)) \\ &\leq |\beta(u)| + |\beta(v)| + \gamma \max_{\sigma\in\Sigma} (s(\tau_\sigma(u)) + s(\tau_\sigma(v))) \\ &\leq F(s)(u) + F(s)(v) \ , \end{aligned}$$

where the last inequality uses subadditivity of the maximum.

To construct bisimulation seminorms for the states of a weighted automaton A we shall study the fixed points of $F_{A,\gamma}$. We start by showing that $F_{A,\gamma}$ has a unique fixed point whenever γ is small enough.

▶ **Theorem 5.** *Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$. If $\gamma < 1/\rho(A)$, then $F_{A,\gamma}$ has a unique fixed point.*

Proof. For simplicity, let $F = F_{A,\gamma}$. By the assumption on γ there exists some $\delta > 0$ such that $\gamma \leq 1/(\rho(A)+\delta)$. Now take $M = \{\tau_\sigma\}_{\sigma\in\Sigma}$ and $\eta = \delta/2$ and let $\|\cdot\|$ be the corresponding quasi-extremal norm on V obtained from Theorem 3. Using this norm we can endow $\mathcal{S}$ with the metric given by $d(s,s') = \sup_{\|v\|\leq 1} |s(v) - s'(v)|$ to obtain a complete metric space $(\mathcal{S}, d)$. To see this, note that for a fixed v with $\|v\| \leq 1$ the sequence $(s_n(v))$ is Cauchy, hence convergent. Call this limit $s(v)$; it is straightforward to see that this defines a seminorm. Thus, if we show that F is a contraction on $\mathcal{S}$ with respect to this metric, then by Banach's fixed point theorem F has a unique fixed point. To see that F is indeed a contraction we start by observing that:

$$d(F(s),F(s')) = \sup_{\|v\|\leq 1} |F(s)(v) - F(s')(v)| = \gamma \sup_{\|v\|\leq 1} \left| \max_\sigma s(\tau_\sigma(v)) - \max_{\sigma'} s'(\tau_{\sigma'}(v)) \right| \ . \quad (2)$$

Fix any $v \in V$ with $\|v\| \leq 1$ and suppose without loss of generality (otherwise we exchange s and s') that $\max_\sigma s(\tau_\sigma(v)) \geq \max_{\sigma'} s'(\tau_{\sigma'}(v))$. Then, letting $\sigma_* = \arg\max_\sigma s(\tau_\sigma(v))$ and using the absolute homogeneity of s and s', it can be shown that:

$$\left| \max_\sigma s(\tau_\sigma(v)) - \max_{\sigma'} s'(\tau_{\sigma'}(v)) \right| \leq \|\tau_{\sigma_*}(v)\| d(s,s') \ . \quad (3)$$

We refer the reader to the appendix [4], for a full derivation. Finally, we use the definition of $\|\cdot\|$ and the choices of δ and η to see that

$$\gamma \|\tau_{\sigma_*}(v)\| \leq \gamma(\rho(A)+\eta)\|v\| \leq \frac{\rho(A)+\delta/2}{\rho(A)+\delta} < 1 \ ,$$

from which we conclude by combining (2) with (3) that $d(F(s),F(s')) < d(s,s')$. ◀

We now exhibit the fixed point of $F_{A,\gamma}$ in closed form. This provides a useful formula for studying properties of the resulting seminorm.

▶ **Theorem 6.** *Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$. Suppose $\gamma < 1/\rho(A)$ and let $s_{A,\gamma} \in \mathcal{S}$ be the fixed point of $F_{A,\gamma}$. Then for any $v \in V$ we have*

$$s_{A,\gamma}(v) = \sup_{x\in\Sigma^\infty} \sum_{t=0}^{\infty} \gamma^t |\beta(\tau_{x_{\leq t}}(v))| = \sup_{x\in\Sigma^\infty} \sum_{t=0}^{\infty} \gamma^t |f_{A_v}(x_{\leq t})| \ . \quad (4)$$

The proof can be found in the appendix [4]. The next theorem is the main result of this section. It shows that any seminorm arising as a fixed point of $F_{A,\gamma}$ captures the notion of A-bisimulation through its kernel for any γ. Namely, two states $u, v \in V$ are A-bisimilar if and only if $s_{A,\gamma}(u - v) = 0$. Note that this result is independent of the choice of γ, as long as the fixed point of $F_{A,\gamma}$ is guaranteed to exist.

▶ **Definition 7.** Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ be a weighted automaton with A-bisimulation $\sim_A$. We say that a seminorm s over V is a *bisimulation seminorm* for A if $\ker(s) = \ker(\sim_A)$.

▶ **Theorem 8.** *Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$. For any $0 < \gamma < 1/\rho(A)$ the fixed point $s_{A,\gamma} \in \mathcal{S}$ of $F_{A,\gamma}$ is a bisimulation seminorm for A.*

Proof. For simplicity, let $F = F_{A,\gamma}$ and $s = s_{A,\gamma}$. Since $W_A = \ker(\sim_A)$ is the largest bisimulation for A, it suffices to show that $\ker(s)$ is a bisimulation for A with $W_A \subseteq \ker(s)$. For the first property we recall that $\ker(s)$ is a linear subspace of V and note that for any $v \in \ker(s)$ we have, using Theorem 6,

$$0 = s(v) = |\beta(v)| + \sup_{x\in\Sigma^\infty} \sum_{t=1}^{\infty} \gamma^t |\beta(\tau_{x_{\leq t}}(v))| \geq |\beta(v)| \geq 0 \ .$$

Therefore $\ker(s) \subseteq \ker(\beta)$. Using the fact that $\beta(v) = 0$, we can also verify the invariance of $\ker(s)$ under all τ_σ, namely $s(\tau_\sigma(v)) = 0$ for all $v \in \ker(s)$ and $\sigma \in \Sigma$ (the full derivation is shown in the appendix [4]). Therefore $\ker(s)$ is a bisimulation for A.

Now let $v \in W_A$. Since W_A is contained in the kernel of β and is invariant for all τ_σ, we see that $\beta(\tau_x(v)) = 0$ for all $x \in \Sigma^\star$. Therefore, using the expression for s given in Theorem 6 we obtain $s(v) = 0$. This concludes the proof. ◀

Because every fixed point of $F_{A,\gamma}$ is a seminorm whose kernel agrees with that of Boreale's bisimulation relation $\sim_A$, we shall call them *γ-bisimulation seminorms* for A. Interestingly, we can show that when A is observable then every γ-bisimulation seminorm is in fact a norm.

▶ **Corollary 9.** *Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ and $\gamma < 1/\rho(A)$. If A is observable then the γ-bisimulation seminorm $s_{A,\gamma}$ is a norm.*

Proof. By Theorem 8 and the observability of A we have $\ker(s_{A,\gamma}) = \ker(\sim_A) = \{0\}$. Thus, $s_{A,\gamma}$ is a norm. ◀

Given an automaton A, and state vectors $v, w \in V$, the pseudometric between states of A induced by $s_{A,\gamma}$ is $d_{A,\gamma}(v, w) = s_{A,\gamma}(v - w)$. Pseudometrics of this form will be called *γ-bisimulation pseudometrics*. By Corollary 9, if A is observable then $d_{A,\gamma}$ is in fact a metric.

To conclude this section we show how to use our γ-bisimulation pseudometrics to define a pseudometric between weighted automata. In order to capture the idea of distance between two WFA let us build the automaton computing the difference between their functions. Given weighted automata $A_i = \langle \Sigma, V_i, \alpha_i, \beta_i, \{\tau_{i,\sigma}\}_{\sigma\in\Sigma}\rangle$ for $i = 1, 2$, we define their *difference automaton* as $A = A_1 - A_2 = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ where $V = V_1 \oplus V_2$, $\alpha = \alpha_1 \oplus (-\alpha_2)$, $\beta = \beta_1 \oplus \beta_2$, and $\tau_\sigma = \tau_{1,\sigma} \oplus \tau_{2,\sigma}$ for all $\sigma \in \Sigma$. Note that A satisfies $f_A(x) = f_{A_1}(x) - f_{A_2}(x)$ for all $x \in \Sigma^\star$ and that $\rho(A) = \max\{\rho(A_1), \rho(A_2)\}$. Then, letting $s_{A,\gamma}$ be the bisimulation seminorm for A we are ready to define our bisimulation distance between weighted automata.

▶ **Definition 10.** Let A_1 and A_2 be two weighted automata and let A be their difference automaton. For any $\gamma < 1/\rho(A)$ we define the *γ-bisimulation distance* between A_1 and A_2 as $d_\gamma(A_1, A_2) = s_{A,\gamma}(\alpha)$.

By exploiting the closed form expression for $s_{A,\gamma}$ given in Theorem 6 we can provide a closed form expression for d_γ.

▶ **Corollary 11.** *Let A_1 and A_2 two weighted automata and $\gamma < 1/\max\{\rho(A_1), \rho(A_2)\}$. Then the γ-bisimulation distance between A_1 and A_2 is given by*

$$d_\gamma(A_1, A_2) = \sup_{x \in \Sigma^\infty} \sum_{t=0}^{\infty} \gamma^t \left| f_{A_1}(x_{\leq t}) - f_{A_2}(x_{\leq t}) \right| \quad . \tag{5}$$

Using the properties of our bisimulation seminorms one can immediately see that d_γ is indeed a pseudometric between all pairs of WFA such that $\gamma < 1/\rho(A_1 - A_2)$. It is also easy to see that d_γ captures the notion of equivalence between weighted automata, in the sense that $d_\gamma(A_1, A_2) = 0$ if and only if $f_{A_1} = f_{A_2}$. Therefore, since minimal weighted automata are unique up to a change of basis, the only way to have $d_\gamma(A_1, A_2) = 0$ when A_1 is minimal is to have either $A_1 = A_2$ or A_2 is a non-minimal WFA recognizing the same weighted language as A_1. In particular, this implies that d_γ is a metric on the set of all minimal WFA A with $\gamma < 1/\rho(A)$.

4 Continuity Properties

In this section we study several continuity properties of our bisimulation pseudometrics between weighted automata. The continuity notions we consider are adapted from those presented by Jaeger et al. in [24], which are developed for labelled Markov chains. Here we extend their definitions of parameter continuity and property continuity to the case of weighted automata. Such notions can be motivated by applications of metrics between transition systems to problems in machine learning [15, 19, 18]; see Section 6 for a discussion on how to use our bisimulation pseudometrics in the analysis of learning algorithms.

4.1 Parameter Continuity

Given a sequence of weighted automata A_i converging to a weighted automaton A, parameter continuity captures the notion that, as the weights in A_i converge to the weights in A, the behavioural distance between A_i and A tends to zero. To make this formal we first define convergence for a sequence of automata and then parameter continuity.

▶ **Definition 12.** Let $(A_i)_{i\in\mathbb{N}}$ be a sequence of WFA $A_i = \langle \Sigma, V, \alpha_i, \beta_i, \{\tau_{i,\sigma}\}_{\sigma\in\Sigma}\rangle$ over the same alphabet Σ and normed vector space $(V, \|\cdot\|)$. We say that the sequence (A_i) *converges* to $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ if $\lim_{i\to\infty} \|\alpha_i - \alpha\| = 0$, $\lim_{i\to\infty} \|\beta_i - \beta\|_* = 0$, and $\lim_{i\to\infty} \|\tau_{i,\sigma} - \tau_\sigma\| = 0$ for all $\sigma \in \Sigma$.

▶ **Definition 13.** A pseudometric d between weighted automata is *parameter continuous* if for any sequence $(A_i)_{i\in\mathbb{N}}$ converging to some weighted automaton A we have $\lim_{i\to\infty} d(A, A_i) = 0$.

The main result of this section is the following theorem stating that our bisimulation pseudometric d_γ is parameter continuous.

▶ **Theorem 14.** *The γ-bisimulation distance between weighted automata is parameter continuous for any sequence of weighted automata $(A_i)_{i\in\mathbb{N}}$ converging to a weighted automaton A with $\gamma < 1/\rho(A)$.*

The proof of this result is quite technical and combines the following two tools:

1. A technical estimate of $d_\gamma(A, A_i)$ in terms of the distance between the weights of A and A_i with respect to a certain norm (Lemma 25 in the appendix [4]). This result also plays a prominent result in Section 6.
2. Several topological properties of the joint spectral radius discussed in Section 2.3.

These proofs are given in the appendix [4].

4.2 Input Continuity

Inspired by the notion of property continuity presented in [24], input g-continuity encapsulates the idea that an upper bound on the behavioural distance between two systems should entail an upper bound on the difference between their outputs on any input $x \in \Sigma^\star$.

▶ **Definition 15.** Let $g : \mathbb{N} \to \mathbb{R}$ be such that $g(l) > 0$ for all $l \in \mathbb{N}$. A distance function d between weighted automata is *input g-continuous* when the following holds: if $(A_i)_{i\in\mathbb{N}}$ is a sequence of weighted automata such that $\lim_{i\to\infty} d(A, A_i) = 0$ for some weighted automaton A, then one has

$$\lim_{i\to\infty} \sup_{x\in\Sigma^\star} \frac{|f_A(x) - f_{A_i}(x)|}{g(|x|)} = 0 \ . \tag{6}$$

Note the special case $g(l) = 1$ is tightly related to the notion of property continuity presented in [24]. The authors of that paper consider differences between the probabilities of the same event under different labelled Markov chains, and therefore always have numbers between 0 and 1. However, for general weighted automata the quantity $|f_A(x) - f_{A'}(x)|$ can grow unboundedly with $|x|$. Thus, in some cases we will need to have a $g(|x|)$ growing with $|x|$ in order to guarantee that (6) stays bounded. The next two results, whose proofs are deferred to the appendix [4], show that essentially $g(|x|) = \gamma^{-|x|}$ is the threshold between input continuity and input non-continuity in our γ-bisimulation pseudometrics.

▶ **Theorem 16.** *The pseudometric d_γ from Definition 10 is input g-continuous for any $g(l) = \Omega(\gamma^{-l})$.*

Note that when $\gamma > 1$ (i.e. when dealing with weighted automata with $\rho(A) \leq 1$) we have $g(l) = 1 \in \Omega(\gamma^{-l})$. This shows that in the case of weighted automata A where every transition operator τ_σ can be represented by a stochastic matrix – a fact that implies $\rho(A) = 1$ – our γ-bisimulation pseudometric is property continuous with respect to the definition in [24].

Further, if g does not grow fast enough as a function of the size of $x \in \Sigma^\star$, then our bisimulation pseudometric is not input g-continuous. In particular, the proof of Theorem 17 provides simple examples of cases where d_γ is not input g-continuous.

▶ **Theorem 17.** *Let $0 < \gamma < 1$. The pseudometric d_γ from Definition 10 is not input g-continuous for any $g(l) = c^{o(l)}$ with $c > 1$.*

5 An Undecidability Result

In this section we will prove that given a weighted automaton $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$, a discount factor $\gamma < 1/\rho(A)$, and a threshold $\nu > 0$, it is undecidable to check whether $s_{A,\gamma}(\alpha) > \nu$. This implies that in general the seminorms and pseudometrics studied in the previous sections are not computable.

The proof of our undecidability result involves a reduction from an undecidable planning problem. Partially observable Markov decision processes (POMDPs) are a generalization of

Markov Decision Processes (MDPs) where we have a set of observations Ω and conditional observation probabilities O. Each state emits some observation $o \in \Omega$ with a certain probability, and so we have a belief over which state we are in after taking an action and observing o. An MDP is a special case of a POMDP where each state has a unique observation, and an unobservable Markov decision process (UMDP) is a special case of a POMDP where all the states emit the same observation. While planning for infinite-horizon UMDPs is undecidable [27], planning for finite-horizon POMDPs is decidable.

Formally, a UMDP is a tuple $U = \langle \Sigma, Q, \alpha, \{\beta_\sigma\}_{\sigma\in\Sigma}, \{T_\sigma\}_{\sigma\in\Sigma}, \gamma\rangle$ where Σ is a finite set of actions, Q is a finite set of states, $\alpha : Q \to [0,1]$ is a probability distribution over initial states in Q, $\beta_\sigma : Q \to \mathbb{R}$ represents the rewards obtained by taking action σ from every state in Q, $T_\sigma : Q \times Q \to [0,1]$ is the transition kernel between states for action σ (i.e. $T_\sigma(q,q')$ is the probability of transitioning to q' given that action σ is taken in q), and $0 < \gamma < 1$ is a discount factor. The value $V_U(x)$ of an infinite sequence of actions $x \in \Sigma^\infty$ in U is the expected discounted cumulative reward collected by executing the actions in x in U starting from a state drawn from α. This can be obtained as follows:

$$V_U(x) = \sum_{t=1}^{\infty} \gamma^{t-1} \alpha^\top T_{x_{\leq t-1}} \beta_{x_t} \ , \tag{7}$$

where $T_y = T_{y_1} \cdots T_{y_t}$ for any finite string $y = y_1 \cdots y_t$ and $T_\epsilon = I$. The following undecidability result was proved by Madani et al. in [27].

▶ **Theorem 18** (Theorem 4.4 in [27]). *The following problem is undecidable: given a UMDP U and a threshold ν decide whether there exists a sequence of actions $x \in \Sigma^\infty$ such that $V_U(x) > \nu$.*

Given a UMDP $U = \langle \Sigma, Q, \alpha, \{\beta_\sigma\}_{\sigma\in\Sigma}, \{T_\sigma\}_{\sigma\in\Sigma}, \gamma\rangle$, we say that U has action-independent rewards if $\beta_\sigma = \beta$ for all $\sigma \in \Sigma$. We say that U has non-negative rewards if $\beta_\sigma(q) \geq 0$ for all $q \in Q$ and $\sigma \in \Sigma$. A careful inspection of the proof in [27] reveals that in fact the reduction provided in the paper always produces as output a UMDP with non-negative action-independent rewards. Thus, we have the following corollary, which forms the basis of our reduction showing that s_γ is not computable.

▶ **Corollary 19.** *The problem in Theorem 18 remains undecidable when restricted to UMDP with non-negative action-independent rewards.*

▶ **Theorem 20.** *The following problem is undecidable: given a weighted automaton $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$, a discount factor $\gamma < 1/\rho(A)$, and a threshold $\nu > 0$, decide whether $s_{A,\gamma}(\alpha) > \nu$.*

Proof. Let $U = \langle \Sigma, Q, \alpha, \beta, \{T_\sigma\}_{\sigma\in\Sigma}, \gamma\rangle$ be a UMDP with non-negative action-independent rewards. With each UMDP of this form we associate the weighted automaton $A = \langle \Sigma, \mathbb{R}^Q, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$. Here we assume that the linear form $\beta : \mathbb{R}^Q \to \mathbb{R}$ is given by $\beta(v) = v^\top \beta$, and that the linear operators $\tau_\sigma : \mathbb{R}^Q \to \mathbb{R}^Q$ are given by $\tau_\sigma(v) = v^\top T_\sigma$.

Note that the matrices T_σ are row-stochastic and therefore we have $\rho(A) \leq \max_\sigma \|\tau_\sigma\|_\infty = 1$. Thus, the discount factor in U satisfies $\gamma < 1 \leq 1/\rho(A)$ and the bisimulation seminorm $s_{A,\gamma}$ associate with A is defined. Using that U has non-negative action-independent rewards we can write for any $x \in \Sigma^\infty$:

$$V_U(x) = \sum_{t=1}^{\infty} \gamma^{t-1} \alpha^\top T_{x_{\leq t-1}} \beta = \sum_{t=0}^{\infty} \gamma^t \alpha^\top T_{x_{\leq t}} \beta = \sum_{t=0}^{\infty} \gamma^t |\alpha^\top T_{x_{\leq t}} \beta| = \sum_{t=0}^{\infty} \gamma^t |\beta(\tau_{x\leq t}(\alpha))| \ .$$

Therefore we have the relation $s_{A,\gamma}(\alpha) = \sup_{x \in \Sigma^\infty} V_U(x)$ between the bisimulation seminorm of A and the value of U. Since deciding whether $V_U(x) > \nu$ for some $x \in \Sigma^\infty$ is undecidable, the theorem follows. ◀

6 Application: Spectral Learning for WFA

An important problem in machine learning is that of finding a weighted automaton $\hat{A}$ approximating an unknown automaton A given only access to data generated by A. A variety of algorithms in different learning frameworks have been considered in the literature; see [5] for an introductory survey. In most learning scenarios it is impossible to exactly recover the target automaton A from a finite amount of data. In that case one aims for algorithms with formal guarantees of the form "the output $\hat{A}$ automaton gets closer to A as the amount of training data grows". To prove such a result one obviously needs a way to measure the distance between two WFA. In this section we show how our γ-bisimulation pseudometric can be used to provide formal learning guarantees for a family of learning algorithms widely referred to as spectral learning. We also briefly discuss the case for behavioural metrics in automata learning problems and compare our metric to other metrics used in the spectral learning literature.

Generally speaking, spectral learning algorithms for WFA work in two phases: the first phase uses the data obtained from the target automaton A to estimate a finite sub-block of the Hankel matrix of f_A; the second phase computes the singular value decomposition of this Hankel matrix and uses the corresponding singular vectors to solve a set of systems of linear equations yielding the weights of the output WFA $\hat{A}$. The Hankel matrix of a function $f : \Sigma^\star \to \mathbb{R}$ is an infinite matrix $H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$ with entries given by $H_f(x, y) = f(xy)$, where xy denotes the string obtained by concatenating the prefix x with the suffix y. Spectral learning algorithms work with a finite sub-block $H \in \mathbb{R}^{P \times S}$ of this Hankel matrix indexed by a set of prefixes $P \subset \Sigma^\star$ and a set of suffixes $S \subset \Sigma^\star$. The pair $B = (P, S)$ is usually an input to the algorithm, in which case formal learning guarantees can be provided under the assumption that B is complete for H_{f_A}. This assumption essentially states that the sub-block of H_{f_A} indexed by B contains enough information to recover a WFA equivalent to A, and is composed of a syntactic condition ensuring B contains a set of prefixes and their extensions by any symbol in Σ, and an algebraic condition ensuring the rank of the Hankel sub-matrix indexed by B has the same rank as the full Hankel matrix H_{f_A}. We refer the reader to [5, 3] for further details about the spectral learning algorithm and a discussion of the completeness property for B. In the sequel we focus on the analysis of the error in the output of the spectral learning algorithm, and show how to provide learning guarantees in terms of our distance d_γ.

The following lemma encapsulates the first step of the analysis of spectral learning algorithms. It shows how the error between the operators of A and $\hat{A}$ depends on the error between the true and the approximated Hankel matrix as measured by the standard operator ℓ_2-norm.

▶ **Lemma 21.** *Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma \in \Sigma} \rangle$ be a WFA and let H be a finite sub-block of the Hankel matrix H_{f_A} indexed by $B = (P, S)$. Suppose $\hat{A} = \langle \Sigma, V, \hat{\alpha}, \hat{\beta}, \{\hat{\tau}_\sigma\}_{\sigma \in \Sigma} \rangle$ is the WFA returned by the spectral learning algorithm using an estimation $\hat{H}$ of H. Let $\|\cdot\|$ be any norm on V. If B is complete, then we have $\|\alpha - \hat{\alpha}\|, \|\beta - \hat{\beta}\|_*, \max_{\sigma \in \Sigma} \|\tau_\sigma - \hat{\tau}_\sigma\| \leq O(\|H - \hat{H}\|_2)$ as $\|H - \hat{H}\|_2 \to 0$. Furthermore, the constants hidden in the big-O notation only depend on the norm $\|\cdot\|$, the Hankel sub-block indices $B = (P, S)$, and the size of the alphabet $|\Sigma|$.*

Proof. Combine Lemma 9.3.5 and Lemma 6.3.2 from [2]. ◀

The results from [2] also provide explicit expressions for the constants hidden in the big-O notation. Concentration of measure for random matrices can be used to show that as the amount of training data increases then the distance between H and $\hat{H}$ converges to zero with high probability (see e.g. [13]). Thus, Lemma 21 implies that as more training data becomes available, spectral learning will output a WFA $\hat{A}$ converging to A.

The last step in the analysis involves showing that as the weights of $\hat{A}$ get closer to the weights of A, the behaviour of the two automata also gets closer. Invoking the parameter continuity of d_γ (Theorem 14) one readily sees that $d_\gamma(A, \hat{A}) \to 0$ as $\|H - \hat{H}\|_2 \to 0$. This provides a proof of consistency of spectral learning with respect to the γ-bisimulation pseudometric. However, machine learning applications often require more precise information about the convergence rate of $d_\gamma(A, \hat{A})$ in order to, for example, compute the amount of data required to achieve a certain error. The following result provides such rate of convergence in the case where the target automaton is irreducible.

▶ **Theorem 22.** *Let $A = \langle \Sigma, V, \alpha, \beta, \{\tau_\sigma\}_{\sigma\in\Sigma}\rangle$ be an irreducible WFA and let H be a finite sub-block of the Hankel matrix H_{f_A} indexed by $B = (P, S)$. Suppose $\hat{A} = \langle \Sigma, V, \hat{\alpha}, \hat{\beta}, \{\hat{\tau}_\sigma\}_{\sigma\in\Sigma}\rangle$ is the WFA returned by the spectral learning algorithm using an estimation $\hat{H}$ of H. Suppose B is complete. Then for any $\gamma < 1/\rho(A)$ we have $d_\gamma(A, \hat{A}) \leq O(\|H - \hat{H}\|_2)$ as $\|H - \hat{H}\|_2 \to 0$. Furthermore, the hidden constants in the big-O notation only depend on A, γ, the Hankel block indices $B = (P, S)$, and the size of the alphabet $|\Sigma|$.*

The local Lipschitz continuity of ρ around irreducible sets of matrices plays an important role in the proof of this result (see the appendix [4]). Nonetheless, the irreducibility constraint is not a stringent one since the sets of irreducible matrices are known to be dense among compact sets of matrices with respect to the Hausdorff metric.

We conclude this section by comparing Theorem 22 with analyses of spectral learning based on other error measures. We start by noting that all finite-sample analyses of spectral learning for WFA we are aware of in the literature provide error bounds in terms of some finite variant of the ℓ_1 distance. In particular, the analyses in [23, 31] bound $\sum_{x\in\Sigma^t} |f_A(x) - f_{\hat{A}}(x)|$ for a fixed $t \geq 0$, while the analyses in [1, 2, 21] extend the bounds to $\sum_{x\in\Sigma^{\leq t}} |f_A(x) - f_{\hat{A}}(x)|$ for a fixed $t \geq 0$. This approach poses several drawbacks, including:

1. Finite ℓ_1-norms provide a pseudo-metric between WFA whose kernel includes pairs of non-equivalent WFA.
2. The number of samples required to achieve a certain error increase with the horizon t, meaning that more data is required to get the same error on longer strings, and that existing bounds become vacuous in the case $t \to \infty$.

In contrast, our result in terms of d_γ establishes a bound on the discrepancy between A and $\hat{A}$ on strings of arbitrary length and will never assign zero distance to a pair of automata realizing different functions. Furthermore, our bisimulation metric still makes sense outside the setting of spectral learning of probabilistic automata where most of the techniques mentioned above have been developed.

7 Conclusion

The metric developed in this paper was very much motivated and informed by spectral ideas. Not surprisingly it was well suited for analyzing spectral learning algorithms for weighted automata. Two obvious directions for future work are:

1. Approximation algorithms for the bisimulation metric.
2. Exploring the relation to approximate minimization.

Both of these are well underway. It seems that some recent ideas from non-linear optimization are very useful in developing approximation algorithms and we hope to be able to report our results soon. Exploring the relation to approximate minimization is less further along, but the spectral ideas at the heart of the approximate minimization algorithm in [6] should be well adapted to the techniques of the present paper.

Acknowledgements. We would like to thank Doina Precup who was actively involved in the approximate minimization work.

References

1. R. Bailly. *Méthodes spectrales pour l'inférence grammaticale probabiliste de langages stochastiques rationnels*. PhD thesis, Aix-Marseille Université, 2011.
2. Borja Balle. *Learning Finite-State Machines: Algorithmic and Statistical Aspects*. PhD thesis, Universitat Politècnica de Catalunya, 2013.
3. Borja Balle, Xavier Carreras, Franco M. Luque, and Ariadna Quattoni. Spectral learning of weighted automata: A forward-backward perspective. *Machine Learning*, 2014.
4. Borja Balle, Pascale Gourdeau, and Prakash Panangaden. Bisimulation metrics for weighted automata, 2017. URL: `https://arxiv.org/abs/1702.08017`.
5. Borja Balle and Mehryar Mohri. Learning weighted automata. In *Conference on Algebraic Informatics*, 2015.
6. Borja Balle, Prakash Panangaden, and Doina Precup. A canonical form for weighted automata and applications to approximate minimization. In *Proceedings of the Thirtieth Annual ACM-IEEE Symposium on Logic in Computer Science*, July 2015.
7. Nikita E. Barabanov. On the Lyapunov indicator of discrete inclusions, part I, II, and III. *Avtomatika i Telemekhanika*, 2:40–46, 1988.
8. Vincent D. Blondel, Yurii Nesterov, and Jacques Theys. On the accuracy of the ellipsoid norm approximation of the joint spectral radius. *Linear Algebra and its Applications*, 394:91–107, 2005.
9. Filippo Bonchi, Marcello M. Bonsangue, Helle Hvid Hansen, Prakash Panangaden, Jan Rutten, and Alexandra Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. *ACM Transactions on Computational Logic*, 2014.
10. Michele Boreale. Weighted bisimulation in linear algebraic form. In *CONCUR 2009 – Concurrency Theory*, pages 163–177. Springer, 2009.
11. Ingrid Daubechies and Jeffrey C. Lagarias. Sets of matrices all infinite products of which converge. *Linear algebra and its applications*, 161:227–263, 1992.
12. Ingrid Daubechies and Jeffrey C. Lagarias. Corrigendum/addendum to: Sets of matrices all infinite products of which converge. *Linear Algebra and its Applications*, 327(1-3):69–83, 2001.
13. François Denis, Mattias Gybels, and Amaury Habrard. Dimension-free concentration bounds on hankel matrices for spectral learning. *Journal of Machine Learning Research*, 17(31):1–32, 2016.
14. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov systems. In *Proceedings of CONCUR99*, number 1664 in Lecture Notes in Computer Science. Springer-Verlag, 1999.
15. Josée Desharnais, Vineet Gupta, Radhakrishnan Jagadeesan, and Prakash Panangaden. A metric for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, June 2004.
16. Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of weighted automata*. EATCS Monographs on Theoretical Computer Science. Springer, 2009.

17 Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*, pages 162–169. AUAI Press, 2004.

18 Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for Markov decision processes with infinite state spaces. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 201–208, July 2005.

19 Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

20 A. Giacalone, C. Jou, and S. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of the Working Conference on Programming Concepts and Methods*, IFIP TC2, 1990.

21 Hadrien Glaude and Olivier Pietquin. PAC learning of Probabilistic Automaton based on the Method of Moments. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 820–829, 2016.

22 Christopher Heil and Gilbert Strang. Continuity of the joint spectral radius: application to wavelets. In *Linear Algebra for Signal Processing*, pages 51–61. Springer, 1995.

23 Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5), 2012.

24 Manfred Jaeger, Hua Mao, Kim Guldstrand Larsen, and Radu Mardare. Continuity properties of distances for Markov processes. In *Proceedings of QEST 2014 Quantitative Evaluation of Systems: 11th International Conference*, pages 297–312. Springer International Publishing, 2014.

25 Raphaël Jungers. *The joint spectral radius: theory and applications*, volume 385. Springer Science and Business Media, 2009.

26 Victor Kozyakin. An explicit lipschitz constant for the joint spectral radius. *Linear Algebra and its Applications*, 433(1):12–18, 2010.

27 Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1-2):5–34, 2003.

28 Prakash Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009.

29 Gian-Carlo Rota and W. Strang. A note on the joint spectral radius. *Indag. Math.*, 22:379–381, 1960.

30 Marcel Paul Schützenberger. On the definition of a family of automata. *Information and control*, 4(2):245–270, 1961.

31 S. M. Siddiqi, B. Boots, and G. Gordon. Reduced-rank hidden Markov models. In *AISTATS*, 2010.

32 Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic systems. In *Proceedings of the Twenty-eighth International Colloquium on Automata, Languages and Programming*. Springer-Verlag, July 2001.

33 Fabian Wirth. The generalized spectral radius and extremal norms. *Linear Algebra and its Applications*, 342(1-3):17–40, 2002.

On the Metric-Based Approximate Minimization of Markov Chains*

Giovanni Bacci[1], Giorgio Bacci[2], Kim G. Larsen[3], and Radu Mardare[4]

1 **Department of Computer Science, Aalborg University, Aalborg, Denmark**
giovbacci@cs.aau.dk
1 **Department of Computer Science, Aalborg University, Aalborg, Denmark**
grbacci@cs.aau.dk
1 **Department of Computer Science, Aalborg University, Aalborg, Denmark**
klg@cs.aau.dk
1 **Department of Computer Science, Aalborg University, Aalborg, Denmark**
mardare@cs.aau.dk

Abstract

We address the behavioral metric-based approximate minimization problem of Markov Chains (MCs), i.e., given a finite MC and a positive integer k, we are interested in finding a k-state MC of *minimal* distance to the original. By considering as metric the bisimilarity distance of Desharnais at al., we show that optimal approximations always exist; show that the problem can be solved as a bilinear program; and prove that its threshold problem is in PSPACE and NP-hard. Finally, we present an approach inspired by expectation maximization techniques that provides suboptimal solutions. Experiments suggest that our method gives a practical approach that outperforms the bilinear program implementation run on state-of-the-art bilinear solvers.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Reducibility and Completeness, I.2.6 Parameter Learning

Keywords and phrases Behavioral distances, Probabilistic Models, Automata Minimization.

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.104

1 Introduction

Minimization of finite automata, i.e., the process of transforming a given finite automaton into an equivalent one with minimum number of states, has been a major subject since the 1950s due to its fundamental importance for any implementation of finite automata tools.

The first algorithm for the minimization of deterministic finite automata (DFAs) is due to Moore [27], with time complexity $O(n^2 s)$, later improved by the now classical Hopcroft's algorithm [17] to $O(ns \log n)$, where n is the number of states and s the size of the alphabet. Their algorithms are based on a partition refinement of the states into equivalence classes of the *Myhill-Nerode equivalence relation*. Partition refinement has been employed in the definition of efficient minimization procedures for a wide variety of automata: by Kanellakis and Smolka [19, 20] for the minimization of labelled transition systems (LTSs) w.r.t. Milner's

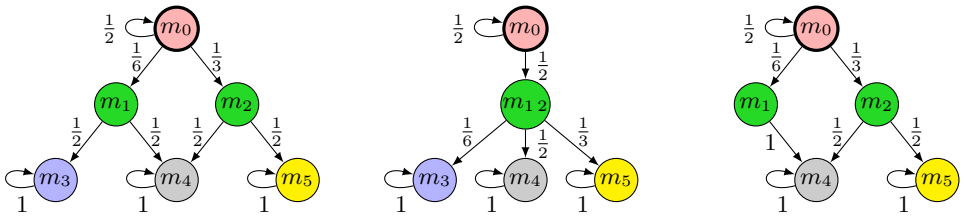
* Work supported by the EU 7th Framework Programme (FP7/2007-13) under Grants Agreement nr.318490 (SENSATION), nr.601148 (CASSTING), the Sino-Danish Basic Research Center IDEA4CPS funded by Danish National Research Foundation and National Science Foundation China, the ASAP Project (4181-00360) funded by the Danish Council for Independent Research, the ERC Advanced Grant LASSO, and the Innovation Fund Denmark center DiCyPS.

strong bisimulation [26]; by Baier [4] for the reduction of Markov Chains (MCs) w.r.t. Larsen and Skou's probabilistic bisimulation [23]; by Alur et al. [2] and by Yannakakis and Lee [30], respectively, for the minimization of timed transition systems and timed-automata. This technique was used also in parallel and distributed implementations of the above algorithms [31, 8], and in the online reachability analysis of transition systems [24].

In [18], Jou and Smolka observed that for reasoning about the behavior of probabilistic systems (and more in general, all type of quantitative systems), rather than equivalences, a notion of distance is more reasonable in practice, since it permits "*a shift in attention from equivalent processes to probabilistically similar processes*". This observation motivated the development of metric-based semantics for quantitative systems, that consists in proposing 1-bounded pseudometrics capturing the similarities of the behaviors in the presence of small variations of the quantitative data. These pseudometrics generalize behavioral equivalences in the sense that, two processes are at distance 0 iff they are equivalent, and at distance 1 if no significant similarities can be observed between them.

The first proposal of a behavioral pseudometric is due to Desharnais et al. [12] on labelled MCs, a.k.a. *probabilistic bisimilarity distance*, with the property that two MCs are at distance 0 iff they are probabilistic bisimilar. Its definition is parametric on a discount factor $\lambda \in (0, 1]$ that controls the significance of the future steps in the measurement. This pseudometric has been greatly studied by van Breugel and Worrell [28, 29, 10] who noticed, among other notable results, its relation with the Kantorovich distance on probability distributions and provided a polynomial-time algorithm for its computation.

The introduction of metric-based semantics motivated the interest in the approximate minimization of quantitative systems. The goal of approximate minimization is to start from a minimal automaton and produce a smaller automaton that is close to the given one in a certain sense. The desired size of the approximating automaton is given as input. Inspired by the aggregation of equivalent states typical of partition refinement techniques, in [15], the approximate minimization problem has been approached by aggregating states having relative smaller distance. An example of this approach on MCs using the λ-bisimilarity distance of Desharnais et al. is shown below.



Let $\mathcal{M}$ be the MC on the left and assume we want to approximate it by an MC with at most 5 states. Since m_1, m_2 are the only two states at distance less than 1, the most natural choice for an aggregation shall collapse (via convex combination) m_1 and m_2, obtaining the MC in the middle, which has distance $\frac{4}{9}(\frac{\lambda^2}{2-\lambda})$ from $\mathcal{M}$. However, the approximate aggregation of states does not necessarily yield the closest optimal solution. Indeed, the MC on the right is a closer approximant of $\mathcal{M}$, at distance $\frac{1}{6}(\frac{\lambda^2}{2-\lambda})$ from it.

In this paper we address the issue of finding *optimal* solutions to the approximate minimization problem. Specifically we aim to answer to the following problem, left open in [15]: "*given a finite MC and a positive integer k, what is its 'best' k-state approximant? Here by 'best' we mean a k-state MC at minimal distance to the original*". We refer to this

problem as *Closest Bounded Approximant* (CBA) and we present the following results related to it.

1. We characterize CBA as a bilinear optimization problem, proving the existence of *optimal* solutions. As a consequence of this result, approximations of optimal solutions can be obtained by checking the feasibility of bilinear matrix inequalities (BMIs) [22, 21].
2. We provide upper- and lower-bound complexity results for the threshold problem of CBA, called *Bounded Approximant* problem (BA), that asks whether there exists a k-state approximant with distance from the original MC bounded by a given rational threshold. We show that BA is in PSPACE and NP-hard. As a corollary we obtain NP-hardness for CBA.
3. We introduce the *Minimum Significant Approximant Bound* (MSAB) problem, that asks what is the minimum size k for an approximant to have some significant similarity to the original MC (i.e., at distance strictly less than 1). We show that this problem is NP-complete when one considers the undiscounted bisimilarity distance.
4. Finally, we present an algorithm for finding suboptimal solutions of CBA that is inspired by Expectation Maximization (EM) techniques [25, 7]. Experiments suggest that our method gives a practical approach that outperforms the bilinear program implementation – state-of-the-art bilinear solvers [21] fails to handle MCs with more than 5 states!

Related Work. In [16], the approximate minimization of MCs is addressed via the notion of *quasi-lumpability*. An MC is quasi-lumpable if the given aggregations of the states can be turned into actual bisimulation-classes by a small perturbation of the transition probabilities. This approach differs from ours since there is no relation to a proper notion of behavioral distance (the approximation is w.r.t. the supremum norm of the difference of the stochastic matrices) and we do not consider any approximate aggregation of states. In [6], Balle et al. consider the approximate minimization of weighted finite automata (WFAs). Their method is via a truncation of a canonical normal form for WFAs that they introduced for the SVD decomposition of infinite Hankel matrices. Both [16] and [6] do not consider the issue of finding the *closest* approximant, which is the main focus of this paper, instead they give upper bounds on the distance from the given model.

2 Markov Chains and Bisimilarity Pseudometric

In this section we introduce the notation and recall the definitions of (discrete-time) *Markov chains* (MCs), *probabilistic bisimilarity* of Larsen and Skou [23], and the *probabilistic bisimilarity pseudometric* of Desharnais et al. [13].

For $R \subseteq X \times X$ an equivalence relation, $X/_R$ denotes its quotient set and $[x]_R$ denotes the R-equivalence class of $x \in X$. $\mathcal{D}(X)$ denotes the set of discrete probability distributions on X, i.e., functions $\mu\colon X \to [0,1]$, s.t. $\mu(X) = 1$, where $\mu(E) = \sum_{x\in E} \mu(x)$ for $E \subseteq X$.

In what follows we fix a countable set L of labels.

▶ **Definition 1** (Markov Chain). A *Markov chain* is a tuple $\mathcal{M} = (M, \tau, \ell)$ consisting of a finite nonempty *set of states* M, a *transition distribution function* $\tau\colon M \to \mathcal{D}(M)$, and a *labelling function* $\ell\colon M \to L$.

Intuitively, if $\mathcal{M}$ is in state m it moves to state m' with probability $\tau(m)(m')$. Labels represent atomic properties that hold in certain states. The set of labels of $\mathcal{M}$ is denoted by $L(\mathcal{M}) = \{\ell(m) \mid m \in M\}$. Hereafter, we use $\mathcal{M} = (M, \tau, \ell)$ and $\mathcal{N} = (N, \theta, \alpha)$ to range over MCs and we refer to their constituents implicitly.

▶ **Definition 2** (Probabilistic Bisimulation [23]). An equivalence relation $R \subseteq M \times M$ is a *probabilistic bisimulation* on $\mathcal{M}$ if whenever $m\ R\ n$, then
1. $\ell(m) = \ell(n)$, and
2. for all $C \in M/_R$, $\tau(m)(C) = \tau(n)(C)$.

Two states $m, n \in M$ are *probabilistic bisimilar w.r.t.* $\mathcal{M}$, written $m \sim_{\mathcal{M}} n$ if they are related by some probabilistic bisimulation on $\mathcal{M}$. In fact, probabilistic bisimilarity is the greatest probabilistic bisimulation.

Any bisimulation R on $\mathcal{M}$ induces a quotient construction, the *R-quotient of $\mathcal{M}$*, denoted $\mathcal{M}/_R = (M/_R, \tau/_R, \ell/_R)$, having R-equivalence classes as states, transition function $\tau/_R([m]_R)([n]_R) = \sum_{u \in [n]_R} \tau(m)(u)$, and labelling function $\ell/_R([m]_R) = \ell(m)$. An MC $\mathcal{M}$ is said *minimal* if it is isomorphic to its quotient w.r.t. probabilistic bisimilarity.

A 1-bounded *pseudometric* on X is a function $d\colon X \times X \to [0,1]$ such that, for any $x, y, z \in X$, $d(x,x) = 0$, $d(x,y) = d(y,x)$, and $d(x,y) + d(y,z) \geq d(x,z)$. 1-bounded pseudometrics on X forms a complete lattice under the point-wise partial order $d \sqsubseteq d'$ iff, for all $x, y \in X$, $d(x,y) \leq d'(x,y)$.

A pseudometric is said to lift an equivalence relation if it enjoys the property that two points are at distance zero iff they are related by the equivalence. A lifting for the probabilistic bisimilarity is provided by the *bisimilarity distance* of Desharnais et al. [13]. Its definition is based on the *Kantorovich (pseudo)metric* on probability distributions over a finite set X, defined as $\mathcal{K}(d)(\mu,\nu) = \min\left\{\int d\,\mathrm{d}\omega \mid \omega \in \Omega(\mu,\nu)\right\}$, where d is a (pseudo)metric on X and $\Omega(\mu,\nu)$ denotes the set of *couplings* for (μ,ν), i.e., distributions $\omega \in \mathcal{D}(X \times X)$ such that, for all $E \subseteq X$, $\omega(E \times X) = \mu(E)$ and $\omega(X \times E) = \nu(E)$.

▶ **Definition 3** (Bisimilarity Distance). Let $\lambda \in (0,1]$. The λ-discounted *bisimilarity pseudometric* on $\mathcal{M}$, denoted by δ_λ, is the least fixed-point of the following functional operator on 1-bounded pseudometrics over M (ordered point-wise)

$$\Psi_\lambda(d)(m,n) = \begin{cases} 1 & \text{if } \ell(m) \neq \ell(n) \\ \lambda \cdot \mathcal{K}(d)(\tau(m),\tau(n)) & \text{otherwise}. \end{cases}$$

The operator Ψ_λ is monotonic, hence, by Tarski fixed-point theorem, δ_λ is well defined.

Intuitively, if two states have different labels δ_λ considers them as "incomparable" (i.e., at distance 1), otherwise their distance is given by the Kantorovich distance w.r.t. δ_λ between their transition distributions. The *discount factor* $\lambda \in (0,1]$ controls the significance of the future steps in the measurement of the distance; if $\lambda = 1$, the distance is said *undiscounted*.

The distance δ_λ has also a characterization based on the notion of coupling structure.

▶ **Definition 4** (Coupling Structure). A function $\mathcal{C}\colon M \times M \to \mathcal{D}(M \times M)$ is a *coupling structure* for $\mathcal{M}$ if for all $m, n \in M$, $\mathcal{C}(m,n) \in \Omega(\tau(m),\tau(n))$.

Intuitively, a coupling structure can be thought of as an MC on the cartesian product $M \times M$, obtained as the probabilistic combination of two copies of $\mathcal{M}$.

Given a coupling structure $\mathcal{C}$ for $\mathcal{M}$ and $\lambda \in (0,1]$, let $\gamma^{\mathcal{C}}_\lambda$ be the least fixed-point of the following operator on $[0,1]$-valued functions $d\colon M \times M \to [0,1]$ (ordered point-wise)

$$\Gamma^{\mathcal{C}}_\lambda(d)(m,n) = \begin{cases} 1 & \text{if } \ell(m) \neq \ell(n) \\ \lambda \int d\,\mathrm{d}\mathcal{C}(m,n) & \text{otherwise}. \end{cases}$$

The function $\gamma^{\mathcal{C}}_\lambda$ is called *λ-discounted discrepancy* of $\mathcal{C}$, and the value $\gamma^{\mathcal{C}}_\lambda(m,n)$ is the λ-discounted probability of hitting from (m,n) a pair of states with different labels in $\mathcal{C}$.

▶ **Theorem 5** (Minimal coupling criterion [10]). *For arbitrary MCs $\mathcal{M}$ and discount factors $\lambda \in (0,1]$, $\delta_\lambda = \min\{\gamma^{\mathcal{C}}_\lambda \mid \mathcal{C}$ coupling structure for $\mathcal{M}\}$.*

Usually, MCs are associated with an initial state to be thought of as their initial configurations. In the rest of the paper when we talk about the distance between two MCs, written $\delta_\lambda(\mathcal{M},\mathcal{N})$, we implicitly refer to the distance between their initial states computed over the disjoint union of their MCs.

3 The Closest Bounded Approximant Problem

In this section we introduce the *Closest Bounded Approximant* problem w.r.t. δ_λ (CBA-λ), and give a characterization of it as a bilinear optimization problem.

▶ **Definition 6** (Closest Bounded Approximant). Let $k \in \mathbb{N}$ and $\lambda \in (0,1]$. The *closest bounded approximant problem* w.r.t. δ_λ for an MC $\mathcal{M}$ is the problem of finding an MC $\mathcal{N}$ with at most k states minimizing $\delta_\lambda(\mathcal{M},\mathcal{N})$.

Clearly, when k is greater than or equal to the number of bisimilarity classes of $\mathcal{M}$, an optimal solution of CBA-λ is the bisimilarity quotient. Therefore, without loss of generality, we will assume $1 \leq k < |M|$ and $\mathcal{M}$ to be minimal. Note that, under these assumptions $\mathcal{M}$ must have at least two nodes with different labels.

Let $\mathrm{MC}(k)$ denote the set of MCs with at most k states and $\mathrm{MC}_A(k)$ its restriction to those using only labels in $A \subseteq L$. Using this notation, the optimization problem CBA-λ on the instance $\langle \mathcal{M}, k\rangle$ can be reformulated as finding an MC $\mathcal{N}^*$ such that

$$\delta_\lambda(\mathcal{M},\mathcal{N}^*) = \min\{\delta_\lambda(\mathcal{M},\mathcal{N}) \mid \mathcal{N} \in \mathrm{MC}(k)\}\ , \tag{1}$$

In general, it is not obvious that for arbitrary instances $\langle \mathcal{M}, k\rangle$ a minimum in (1) exists. At the end of the section, we will show that such a minimum always exists (Corollary 9).

A useful property of CBA-λ is that an optimal solution can be found among the MCs using labels from the given MC.

▶ **Lemma 7** (Meaningful labels). *Let $\mathcal{M}$ be an MC. Then, for any $\mathcal{N}' \in \mathrm{MC}(k)$ there exists $\mathcal{N} \in \mathrm{MC}_{L(\mathcal{M})}(k)$ such that $\delta_\lambda(\mathcal{M},\mathcal{N}) \leq \delta_\lambda(\mathcal{M},\mathcal{N}')$.*

In the following, fix $\langle \mathcal{M}, k\rangle$ as instance of CBA-λ, let $m_0 \in M$ be the initial state of $\mathcal{M}$. By Lemma 7, Theorem 5 and Tarski fixed-point theorem

$$\begin{aligned}
&\inf\{\delta_\lambda(\mathcal{M},\mathcal{N}) \mid \mathcal{N} \in \mathrm{MC}(k)\} = && (2)\\
&\quad = \inf\{\gamma^{\mathcal{C}}_\lambda(\mathcal{M},\mathcal{N}) \mid \mathcal{N} \in \mathrm{MC}_{L(\mathcal{M})}(k) \text{ and } \mathcal{C} \in \Omega(\mathcal{M},\mathcal{N})\} && (3)\\
&\quad = \inf\{d(\mathcal{M},\mathcal{N}) \mid \mathcal{N} \in \mathrm{MC}_{L(\mathcal{M})}(k), \mathcal{C} \in \Omega(\mathcal{M},\mathcal{N}), \text{ and } \Gamma^{\mathcal{C}}_\lambda(d) \sqsubseteq d\}\ , && (4)
\end{aligned}$$

where $\Omega(\mathcal{M},\mathcal{N})$ denotes the set of all coupling structures for the disjoint union of $\mathcal{M}$ and $\mathcal{N}$. This simple change in perspective yields a translation of the problem of computing the optimal value of CBA-λ to the bilinear program in Figure 1.

In our encoding, $N = \{n_0, \ldots, n_{k-1}\}$ are the states of an arbitrary $\mathcal{N} = (N, \theta, \alpha) \in \mathrm{MC}(k)$ and n_0 is the initial one. The variable $\theta_{n,v}$ is used to encode the transition probability $\theta(n)(v)$. Hence, a feasible solution satisfying (9–11) will have the variable $c^{m,n}_{u,v}$ representing the value $\mathcal{C}(m,n)(u,v)$ for a coupling structure $\mathcal{C} \in \Omega(\mathcal{M},\mathcal{N})$. An assignment for the variables $\alpha_{n,l}$ satisfying (7–8) encodes (uniquely) a labeling function $\alpha\colon N \to L(\mathcal{M})$ satisfying the following property:

$$\text{for all } n \in N, l \in L(\mathcal{M}) \qquad \alpha_{n,l} = 1 \quad \text{iff} \quad \alpha(n) = l\,. \tag{12}$$

$$
\begin{aligned}
&\text{mimimize} && d_{m_0,n_0} && && \\
&\text{such that} && \lambda \textstyle\sum_{(u,v)\in M\times N} c_{u,v}^{m,n} \cdot d_{u,v} \le d_{m,n} && m\in M,\, n\in N && (5)\\
& && 1-\alpha_{n,l} \le d_{m,n} \le 1 && n\in N,\, l\in L(\mathcal{M}),\, \ell(m)\neq l && (6)\\
& && \alpha_{n,l}\cdot\alpha_{n,l'} = 0 && n\in N,\, l,l'\in L(\mathcal{M}),\, l\neq l' && (7)\\
& && \textstyle\sum_{l\in L(\mathcal{M})} \alpha_{n,l} = 1 && n\in N && (8)\\
& && \textstyle\sum_{v\in N} c_{u,v}^{m,n} = \tau(m)(u) && m,u\in M,\, n\in N && (9)\\
& && \textstyle\sum_{u\in M} c_{u,v}^{m,n} = \theta_{n,v} && m\in M,\, n,v\in N && (10)\\
& && c_{u,v}^{m,n} \ge 0 && m,u\in M,\, n,v\in N && (11)
\end{aligned}
$$

Figure 1 Characterization of CBA-λ as a bilinear optimization problem.

The constraint (7) models the fact that each node $n \in N$ is assigned to at most one label $l \in L(\mathcal{M})$, and the constraint (8) ensures that each node is assigned to at least one label. Conversely, any labeling $\alpha\colon N \to L(\mathcal{M})$ admits an assignment of the variables $\alpha_{n,l}$ that satisfy (7–8) and (12). Finally, an assignment for the variables $d_{m,n}$ satisfying the constraints (5–6) represents a prefix point of $\Gamma_\lambda^{\mathcal{C}}$. Note that (6) guarantees that $d_{m,n} = 1$ whenever $\alpha(n) \neq \ell(m)$ – indeed, by (7), $\alpha_{n,l} = 0$ iff $\alpha(n) \neq \ell(m)$.

Let $F_\lambda\langle\mathcal{M}, k\rangle$ denote the bilinear optimization problem in Figure 1. Directly from the arguments stated above we obtain the following result.

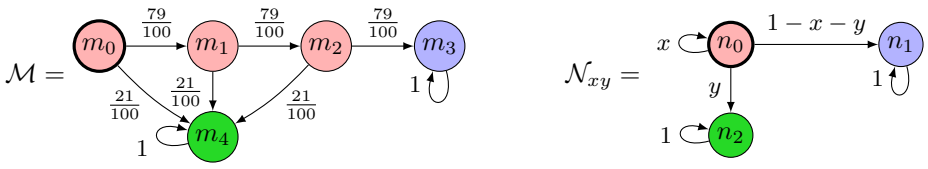
▶ **Theorem 8.** $\inf\{\delta_\lambda(\mathcal{M},\mathcal{N}) \mid \mathcal{N}\in \mathrm{MC}(k)\}$ *is the optimal value of* $F_\lambda\langle\mathcal{M},k\rangle$.

▶ **Corollary 9.** *Any instance of CBA-*λ *admits an optimal solution.*

Proof. Let h be the number of variables in $F_\lambda\langle\mathcal{M},k\rangle$. The constraints (6–11) describe a compact subset of $\mathbb{R}^h$ – it is an intersection of closed sets bounded by $[0,1]^h$. The objective function of $F_\lambda\langle\mathcal{M},k\rangle$ is linear, hence the infimum is attained by a feasible solution. The thesis follows by Theorem 8. ◀

The following example shows that even by starting with a MC with rational transition probabilities, optimal solutions for CBA-λ may have irrational transition probabilities.

▶ **Example 10.** Consider the MC $\mathcal{M}$ depicted below, with initial state m_0 and labeling represented by colors. An optimal solution of CBA-1 on $\langle\mathcal{M},3\rangle$ is the MC $\mathcal{N}_{xy}$ depicted below, with initial state n_0 and parameters $x = \frac{1}{30}\left(10+\sqrt{163}\right)$, $y = \frac{21}{100}$.



Since the distance $\delta_1(\mathcal{M},\mathcal{N}_{xy}) = \frac{436}{675} - \frac{163\sqrt{163}}{13500} \approx 0.49$ is irrational, by [10, Proposition 13], any optimal solution must have some irrational transition probability.

Next we show that the above is indeed an optimal solution. Assume by contradiction that $\mathcal{N}^* \not\sim \mathcal{N}_{xy}$ is an optimal solution. By Lemma 7, we can assume $L(\mathcal{N}^*) \subseteq L(\mathcal{M})$. If

$L(\mathcal{N}^*) = L(\mathcal{M})$, then $\delta_1(\mathcal{M},\mathcal{N}^*) = \min\{\delta_1(\mathcal{M},\mathcal{N}_{zy}) \mid z \in [0,1-y]\}$ since one can show that for any $y' \neq y$ and z', there exists $z \in [0,1-y]$, such that $\delta_1(\mathcal{M},\mathcal{N}_{zy}) \leq \delta_1(\mathcal{M},\mathcal{N}_{z'y'})$. $\delta_1(\mathcal{M},\mathcal{N}_{zy})$ is analytically solved by $z^3 - z^2 - \frac{21}{100}z - \frac{79}{100}$ and its minimum value is achieved at $z = \frac{1}{30}\left(10 + \sqrt{163}\right)$. This contradicts $\mathcal{N}^* \not\sim \mathcal{N}_{xy}$. Assume $L(\mathcal{N}^*) \subsetneq L(\mathcal{M})$. By [10, Corollary 11], for any measurable set $A \subseteq L^\omega$, $\delta_1(\mathcal{M},\mathcal{N}^*) \geq |\mathbb{P}_{\mathcal{M}}(A) - \mathbb{P}_{\mathcal{N}^*}(A)|$, where $\mathbb{P}_{\mathcal{N}}(A)$ denotes the probability that a run of $\mathcal{N}$ is in A. If $\ell(m_0) \notin L(\mathcal{N}^*)$, we have that $\delta_1(\mathcal{M},\mathcal{N}^*) \geq |\mathbb{P}_{\mathcal{M}}(\ell(m_0)L^\omega) - \mathbb{P}_{\mathcal{N}^*}(\ell(m_0)L^\omega)| = \mathbb{P}_{\mathcal{M}}(\ell(m_0)L^\omega) = 1 > \delta_1(\mathcal{M},\mathcal{N}_{xy})$. Analogously, if $\ell(m_3) \notin L(\mathcal{N}^*)$ we have $\delta_1(\mathcal{M},\mathcal{N}^*) \geq \mathbb{P}_{\mathcal{M}}(L^*\ell(m_3)L^\omega) = \left(\frac{79}{100}\right)^3 > \delta_1(\mathcal{M},\mathcal{N}_{xy})$. Finally, if $\ell(m_4) \notin L(\mathcal{N}^*)$, $\delta_1(\mathcal{M},\mathcal{N}^*) \geq \mathbb{P}_{\mathcal{M}}(L^*\ell(m_4)L^\omega) = \frac{21}{100}\sum_{i=0}^{2}\left(\frac{79}{100}\right)^i > \delta_1(\mathcal{M},\mathcal{N}_{xy})$. ◀

4 The Bounded Approximant Threshold Problem

The *Bounded Approximant problem* w.r.t. δ_λ (BA-λ) is the threshold decision problem of CBA-λ, that, given MC $\mathcal{M}$, integer $k \geq 1$, and rational $\epsilon \geq 0$, asks whether there exists $\mathcal{N} \in \mathrm{MC}(k)$ such that $\delta_\lambda(\mathcal{M},\mathcal{N}) \leq \epsilon$.

From the characterization of CBA-λ as a bilinear optimization problem (Theorem 8) we immediately get the following complexity upper-bound for BA-λ.

▶ **Theorem 11.** *For any $\lambda \in (0,1]$, BA-λ is in PSPACE.*

Proof. By Theorem 8, deciding an instance $\langle \mathcal{M}, k, \epsilon\rangle$ of BA-λ can be encoded as a decision problem for the existential theory of the reals, namely, checking the feasibility of the constraints (6–11) in conjunction with $d_{m_0,n_0} \leq \epsilon$. The encoding is polynomial in the size of $\langle \mathcal{M}, k, \epsilon\rangle$, thus it can be solved in PSPACE (*cf.* Canny [9]). ◀

In the rest of the section we provide a complexity lower-bound for BA-λ, by showing that BA-λ is NP-hard via a reduction from Vertex Cover. Recall that, a vertex cover of an undirected graph G is a subset C of vertices such that every edge in G has at least one endpoint in C. Given a graph G and a positive integer h, the Vertex Cover problem asks if G has a cover of size at most h.

Before presenting the reduction we establish structural properties for an optimal solution of CBA-λ in the case the given MC has injective labeling (i.e., no two distinct states with the same label). Specifically, we show that an optimal solution for an instance $\langle \mathcal{M}, k\rangle$ of CBA-λ can be found among MCs with injective labeling into $L(\mathcal{M})$.

▶ **Lemma 12.** *If $\mathcal{M}$ has injective labeling, there exists $\mathcal{N} \in \mathrm{MC}_{L(\mathcal{M})}(k)$ with injective labeling that minimizes the distance $\delta_\lambda(\mathcal{M},\mathcal{N})$.*

▶ **Lemma 13.** *For all $m \in M$ and $n \in N$, $\delta_\lambda(m,n) \geq \lambda \cdot \tau(m)(\{u \in M \mid \ell(u) \notin L(\mathcal{N})\})$.*

Note that Lemma 13 provides a lower-bound on the optimal distance between $\mathcal{M}$ and any $\mathcal{N} \in \mathrm{MC}(k)$. This lower-bound will be useful in the proof of the following result.

▶ **Theorem 14.** *For any $\lambda \in (0,1]$, BA-λ is NP-hard.*

Proof. We provide a polynomial-time many-one reduction from Vertex Cover.

Let $\langle G = (V,E), h\rangle$ be an instance of Vertex Cover and let $e = |E|$. Without loss of generality we assume $e \geq 2$ and $k < n$. From G we construct the MC $\mathcal{M}_G = (M, \tau, \ell)$ as follows. The set of states M is given as the union of V and E to which we add two extra states: a root r (thought of as the initial state) and a sink s. Each node of $\mathcal{M}_G$ is associated with a unique label (i.e., ℓ is injective). The sink state s and all $v \in V$ loop to themselves

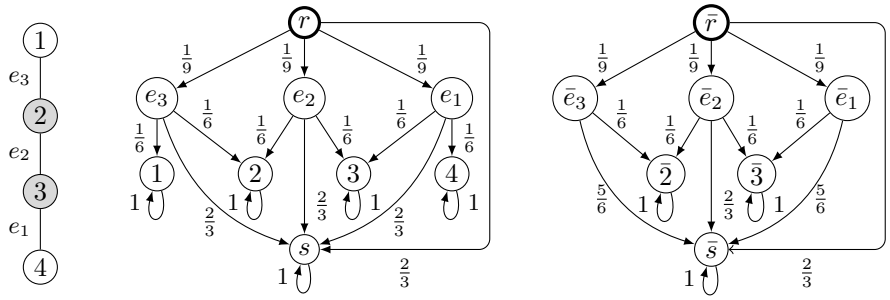


Figure 2 (Left) An undirected graph G; (Center) The MC $\mathcal{M}_G$ associated to the graph G; (Right) The MC $\mathcal{M}_C$ associated to the vertex cover $C = \{2, 3\}$ of G. (see Thm. 14).

with probability 1. All the other states go with probability $1 - \frac{1}{e}$ to the sink state s. The rest of their transition probability mass is assigned as follows. The root r goes with probability $\frac{1}{e^2}$ to each $a \in E$, and all $(u, v) \in E$ go with probability $\frac{1}{2e}$ to their endpoints u, v. An example of construction for $\mathcal{M}_G$ is given in Figure 2. Next we show that

$$\langle G, h\rangle \in \textsc{Vertex Cover} \qquad \text{iff} \qquad \langle \mathcal{M}_G, e + h + 2, \tfrac{\lambda^2}{2e^2}\rangle \in \text{BA-}\lambda\,.$$

($\Rightarrow$) Let C be a h-vertex cover of G. Construct $\mathcal{M}_C \in \mathrm{MC}(e + h + 2)$ by taking a copy of $\mathcal{M}_G$, removing all states in $V \setminus C$, and redirecting the exceeding transition probability to the sink state s (an example is shown in Figure 2). Next we show that $\delta_\lambda(\mathcal{M}_G, \mathcal{M}_C) \leq \frac{\lambda^2}{2e^2}$. For convenience, the states in $\mathcal{M}_C$ will be marked with a bar. By construction of $\mathcal{M}_G, \mathcal{M}_C$, for each $a \in E$, $\delta_\lambda(a, \overline{a}) \leq \frac{\lambda}{2e}$. Thus, $\delta_\lambda(\mathcal{M}_G, \mathcal{M}_C) = \delta_\lambda(r, \bar{r}) = \frac{\lambda}{e^2}\sum_{a\in E}\delta_\lambda(a, \overline{a}) \leq \frac{\lambda^2}{2e^2}$.

($\Leftarrow$) By contradiction, assume there exists $\mathcal{N} = (N, \theta, \alpha) \in \mathrm{MC}(e + h + 2)$ such that $\delta_\lambda(\mathcal{M}_G, \mathcal{N}) \leq \frac{\lambda^2}{2e^2}$ but no vertex cover of G of size h. Since ℓ is injective, by Lemma 12 we can assume α to be injective and $L(\mathcal{N}) \subseteq L(\mathcal{M}_G)$. We consider three cases separately:

Case: $\ell(s) \notin L(\mathcal{N})$. By Lemma 13 and the fact that $e > 1$ and $\lambda \in (0, 1]$, we get the following contradiction: $\delta_\lambda(\mathcal{M}_G, \mathcal{N}) = \delta_\lambda(r, n_0) \geq \lambda \cdot \tau(r)(s) = \frac{\lambda(e-1)}{e} > \frac{\lambda^2}{2e^2}$.

Case: $\ell((u, v)) \notin L(\mathcal{N})$, for some $(u, v) \in E$. By Lemma 13 and the fact that $\lambda \in (0, 1]$ and $e > 1$, leading to the contradiction $\delta_\lambda(\mathcal{M}_G, \mathcal{N}) = \delta_\lambda(r, n_0) \geq \lambda \cdot \tau(r)((u, v)) = \frac{\lambda}{e^2} > \frac{\lambda^2}{2e^2}$.

Case: $\ell(s) \in L(\mathcal{N})$ and $\{\ell((u, v)) \mid (u, v) \in E\} \subseteq L(\mathcal{N})$. Let $N' \subseteq N$ be the states with labels in $\{\ell(u) \mid u \in V\}$. By the structural hypothesis assumed on $\mathcal{N}$, we have $|N'| \leq h$. For each $(u, v) \in E$, two possible cases apply: if $\alpha(n) \in \{\ell(u), \ell(v)\}$, for some $n \in N'$, then $\delta_\lambda((u, v), \overline{(u, v)}) \geq \frac{\lambda}{2e}$; otherwise $\delta_\lambda((u, v), \overline{(u, v)}) \geq \frac{\lambda}{e} > \frac{\lambda}{2e}$. By hypothesis, there is no vertex cover of size h, hence there is at least one edge $(u, v) \in E$ for which the second case applies. Therefore, $\delta_\lambda(\mathcal{M}_G, \mathcal{N}) = \delta_\lambda(r, n_0) = \frac{\lambda}{e^2}\sum_{(u,v)\in E}\delta_\lambda((u, v), \overline{(u, v)}) > \frac{\lambda}{e^2} \cdot e \cdot \frac{\lambda}{2e} = \frac{\lambda^2}{2e^2}$.

The instance $\langle \mathcal{M}_G, e + h + 2, \frac{\lambda^2}{2e^2}\rangle$ of BA-λ can be constructed in polynomial time in the size of $\langle G, h\rangle$. Thus, since Vertex Cover is NP-hard, so is BA-λ. ◀

5 Minimum Significant Approximant Bound

Recall that, two MCs are at distance 1 from each other when there is no significant similarity between their behaviors. Thus an MC $\mathcal{N}$ is said to be a *significant approximant* for the MC $\mathcal{M}$ w.r.t. δ_λ if $\delta_\lambda(\mathcal{M}, \mathcal{N}) < 1$.

Given an MC $\mathcal{M}$, the *Minimum Significant Approximant Bound* problem w.r.t. δ_λ (MSAB-λ) looks for the smallest k such that $\delta_\lambda(\mathcal{M}, \mathcal{N}) < 1$, for some $\mathcal{N} \in \mathrm{MC}(k)$. The decision version of this problem is called *Significant Bounded Approximant problem* w.r.t. δ_λ

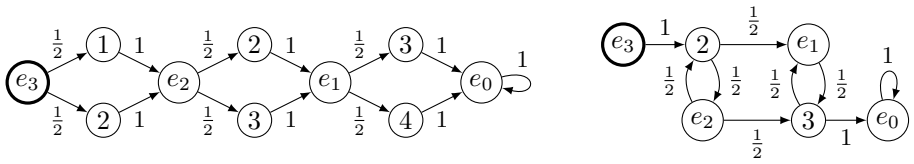


Figure 3 (Left) The MC $\mathcal{M}_G$ associated to the graph G in Figure 2 and (right) an MC $\mathcal{N}$ associated to the vertex cover $C = \{1, 2\}$ of G such that $\delta_1(\mathcal{M}_G, \mathcal{N}) < 1$ (cf. Theorem 16).

(SBA-λ), and asks whether, for a given positive integer k, there exists $\mathcal{N} \in \mathrm{MC}(k)$ such that $\delta_\lambda(\mathcal{M}, \mathcal{N}) < 1$.

When the discount factor $\lambda < 1$, the two problems above turn out to be trivial. Indeed, $\delta_\lambda(\mathcal{M}, \mathcal{N}) \leq \lambda$ when the initial states of $\mathcal{M}$ and $\mathcal{N}$ have the same label. On the contrary, in the case the distance is undiscounted ($\lambda = 1$), these problems are NP-complete. Before presenting the result, we provide the following technical lemma.

▶ **Lemma 15.** *Let $\mathcal{M}$ be a MC (assumed to be minimal) with initial state m_0 and $G(\mathcal{M})$ its underlying directed graph. Then, $\langle \mathcal{M}, k \rangle \in$ SBA-1 iff there exists a bottom strongly connected component (SCC) $G' = (V, E)$ in $G(\mathcal{M})$ and a path $m_0 \dots m_h$ in $G(\mathcal{M})$ such that $m_h \in V$ and $|\{\ell(m_i) \mid i < h, \nexists \text{ a path } v_i \dots v_{h-1} m_h \text{ in } G' \text{ s.t. } \forall i \leq j < h. \ell(m_j) = \ell(v_j)\}| + |V| \leq k$.*

▶ **Theorem 16.** *SBA-1 is NP-complete.*

Proof. The membership in NP is easily proved by using the characterization in Lemma 15 and exploiting Tarjan's algorithm for generating bottom SCCs. As for the NP-hardness, we provide a polynomial-time many-one reduction from Vertex Cover. Let $G = (V, E)$ be a graph with $E = \{e_1, \dots, e_n\}$. We construct the MC $\mathcal{M}_G$ as follows. The set of states is given by the set of edges E along with two states e_i^1 and e_i^2, for each edge $e_i \in E$, representing the two endpoints of e_i and an extra sink state e_0. The initial state is e_n. The transition probabilities are given as follows. The sink state e_0 loops with probability 1 to itself. Each edge $e_i \in E$ goes with probability $\frac{1}{2}$ to e_i^1 and e_i^2, respectively. For $1 \leq i \leq n$, the states e_i^1 and e_i^2 go with probability 1 to the state e_{i-1}. The edge states and the sink state are labelled by pairwise distinct labels, while the endpoints states e_i^1 and e_i^2 are labelled by the node in V they represent. An example of construction for $\mathcal{M}_G$ is shown in Figure 3.

Next we show the following equivalence:

$$\langle G, h \rangle \in \text{Vertex Cover} \qquad \text{iff} \qquad \langle \mathcal{M}_G, h + n + 1 \rangle \in \text{SBA-1} \tag{13}$$

By construction, $\mathcal{M}_G$ is minimal and its underlying graph H has a unique bottom strongly connected component, namely the self-loop in e_0. Each path $p = e_n \rightsquigarrow e_0$ in H passes through all edge states, and the set of labels of the endpoint states in p is a vertex cover of G. Since $e_0, \dots, e_n$ have pairwise distinct labels, we have that G has a vertex cover of size at most h iff there exists a path in H from e_n to e_0 that has at most $n + 1 + h$ different labels. Thus, (13) follows by Lemma 15. ◀

6 An Expectation Maximization-like Heuristic

In this section we describe an approximation algorithm for determining suboptimal solutions of CBA-λ for an arbitrary instance $\langle \mathcal{M}, k \rangle$.

Algorithm 1 Approximate Minimization – Expectation Maximization-like heuristic

Input: $\mathcal{M} = (M, \tau, \ell)$, $\mathcal{N}_0 = (N, \theta_0, \alpha)$, and $h \in \mathbb{N}$.

1. $i \leftarrow 0$
2. **repeat**
3. $\quad i \leftarrow i + 1$
4. $\quad$ compute $\mathcal{C} \in \Omega(\mathcal{M}, \mathcal{N}_{i-1})$ such that $\delta_\lambda(\mathcal{M}, \mathcal{N}_{i-1}) = \gamma^{\mathcal{C}}_\lambda(\mathcal{M}, \mathcal{N}_{i-1})$
5. $\quad \theta_i \leftarrow \textsc{UpdateTransition}(\theta_{i-1}, \mathcal{C})$
6. $\quad \mathcal{N}_i \leftarrow (N, \theta_i, \alpha)$
7. **until** $\delta_\lambda(\mathcal{M}, \mathcal{N}_i) > \delta_\lambda(\mathcal{M}, \mathcal{N}_{i-1})$ or $i \geq h$
8. **return** $\mathcal{N}_{i-1}$

Given an initial approximant $\mathcal{N}_0 \in \mathrm{MC}(k)$, the algorithm produces a sequence of MCs $\mathcal{N}_0, \mathcal{N}_1, \ldots$ in $\mathrm{MC}(k)$ having successively decreased distance from $\mathcal{M}$. We defer until later a discussion of how the initial MC $\mathcal{N}_0$ is chosen. The procedure is described in Algorithm 1.

The intuitive idea of the algorithm is to iteratively update the initial MC by assigning relatively greater probability to transitions that are most representative of the behavior of the MC $\mathcal{M}$ w.r.t. δ_λ. The procedure stops when the last iteration has not yield an improved approximant w.r.t. the preceding one. The input also includes a parameter $h \in \mathbb{N}$ that bounds the number of iterations.

The rest of the section explains two heuristics for implementing the UpdateTransition function invoked at line 5. This function shall return the transition probabilities for the successive approximant (see line 6).

Define $\beta^{\mathcal{C}}_\lambda$ to be the least fixed-point of the following functional operator on 1-bounded real-valued functions $d\colon M \times N \to [0,1]$ (ordered point-wise):

$$B^{\mathcal{C}}_\lambda(d)(m,n) = \begin{cases} 1 & \text{if } \gamma^{\mathcal{C}}_\lambda(m,n) = 0 \\ 0 & \text{if } \ell(m) \neq \alpha(n) \\ (1-\lambda) + \lambda \int_{M\times N} d \, \mathrm{d}\mathcal{C}(m,n) & \text{otherwise}\,. \end{cases}$$

By Theorem 5, the relation $R_{\mathcal{C}} = \big\{(m,n) \mid \gamma^{\mathcal{C}}_\lambda(m,n) = 0\big\}$ is easily shown to be a bisimulation, specifically, the greatest bisimulation induced by $\mathcal{C}$.

Define $\mathcal{C}_\lambda$ as the MC obtained by augmenting $\mathcal{C}$ with an 'sink' state $\perp$ to which any other state moves with probability $(1-\lambda)$. Intuitively, the value $\beta^{\mathcal{C}}_\lambda(m,n)$ can be interpreted as the reachability probability in $\mathcal{C}_\lambda$ of either hitting the sink state or a pair of bisimilar states in $R_{\mathcal{C}}$ along a path formed only by pairs of states with identical labels starting from (m,n).

▶ **Lemma 17.** *For all $m \in M$ and $n \in N$, $\beta^{\mathcal{C}}_\lambda(m,n) = 1 - \gamma^{\mathcal{C}}_\lambda(m,n)$.*

From equation (3) and Lemma 17, we can turn the problem CBA-λ as

$$\operatorname{argmax} \big\{\beta^{\mathcal{C}}_\lambda(\mathcal{M},\mathcal{N}) \mid \mathcal{N} \in \mathrm{MC}_{L(\mathcal{M})}(k),\, \mathcal{C} \in \Omega(\mathcal{M},\mathcal{N})\big\}\,. \tag{14}$$

Equation (14) says that a solution of CBA-λ is the right marginal of a coupling structure $\mathcal{C}$ such that $\mathcal{C}_\lambda$ maximizes the probability of generating paths with prefix in $\cong^*(R_{\mathcal{C}} \cup \perp)$ starting from the pair (m_0, n_0) of initial states[1], where $\cong = \{(m,n) \notin R_{\mathcal{C}} \mid \ell(m) = \alpha(n)\}$.

In the rest of the section we assume $\mathcal{N}_{i-1} \in \mathrm{MC}(k)$ to be the current approximant with associated coupling structure $\mathcal{C} \in \Omega(\mathcal{M}, \mathcal{N}_{i-1})$ as in line 4 in Algorithm 1.

[1] We borrowed notation from regular expressions, such as union, concatenation, and Kleene star, to express the set of finite paths $\cong^* R_{\mathcal{C}}$ as a language over the alphabet $M \times N$.

The "Averaged Marginal" Heuristic. The first heuristic is inspired by the Expectation Maximization (EM) algorithm described in [7]. The idea is to count the expected number of occurrences of the transitions in $\mathcal{C}$ in the set of paths $\cong^* R_{\mathcal{C}}$ and, in accordance with (14), updating $\mathcal{C}$ by increasing the probability of the transitions that were contributing the most.

For each $m, u \in M$ and $n, v \in N$ let $Z^{m,n}_{u,v} \colon (M \times N)^\omega \to \mathbb{N}$ be the random variable that counts the number of occurrences of the edge $((m,n)(u,v))$ in a prefix in $\cong^*(R_{\mathcal{C}} \cup \bot)$ of the given path. We denote by $\mathbf{E}[Z^{m,n}_{u,v} | \mathcal{C}]$ the expected value of $Z^{m,n}_{u,v}$ w.r.t. the probability distribution induced by $\mathcal{C}_\lambda$. Using these values we define the optimization problem EM$\langle \mathcal{N}, \mathcal{C} \rangle$:

$$\begin{aligned}
&\text{maximize} && \textstyle\sum_{m,u \in M} \sum_{n,v \in N} \mathbf{E}[Z^{m,n}_{u,v} | \mathcal{C}] \cdot \ln(c^{m,n}_{u,v}) && \\
&\text{such that} && \textstyle\sum_{v \in N} c^{m,n}_{u,v} = \tau(m)(u) && m,u \in M,\ n \in N && (15)\\
& && \textstyle\sum_{u \in M} c^{m,n}_{u,v} = \theta_{n,v} && m \in M,\ n,v \in N && (16)\\
& && c^{m,n}_{u,v} \geq 0 && m,u \in M,\ n,v \in N &&
\end{aligned}$$

A solution of EM$\langle \mathcal{N}, \mathcal{C} \rangle$ can be used to improve a pair $\langle \mathcal{N}, \mathcal{C} \rangle$ in the sense of (14).

▶ **Theorem 18.** *If $\beta^{\mathcal{C}}_\lambda(\mathcal{M}, \mathcal{N}) > 0$, then an optimal solution for EM$\langle \mathcal{N}, \mathcal{C} \rangle$ describes an MC $\mathcal{N}' \in \mathrm{MC}(k)$ and a coupling structure $\mathcal{C}' \in \Omega(\mathcal{M}, \mathcal{N}')$ such that $\beta^{\mathcal{C}'}_\lambda(\mathcal{M}, \mathcal{N}') \geq \beta^{\mathcal{C}}_\lambda(\mathcal{M}, \mathcal{N})$.*

Unfortunately, EM$\langle \mathcal{N}, \mathcal{C} \rangle$ does not have an easy analytic solution and turns out to be inefficiently solved by nonlinear optimization methods. On the contrary, the relaxed optimization problem obtained by dropping the constraints (16) has a simple analytic solution, and the first heuristic at line 5, updates θ_i as follows[2]

$$c^{m,n}_{u,v} = \frac{\tau(m)(n) \cdot \mathbf{E}[Z^{m,n}_{u,v} | \mathcal{C}]}{\sum_{x \in N} \mathbf{E}[Z^{m,n}_{u,x} | \mathcal{C}]}, \quad \theta_i(n)(v) = \begin{cases} \theta_{i-1}(n)(v) & \text{if } \exists m \in M.\, n\ R_{\mathcal{C}}\ m \\ \dfrac{\sum_{m,u \in M} c^{m,n}_{u,v}}{\sum_{x \in N} \sum_{m,u \in M} c^{m,n}_{u,x}} & \text{otherwise} \end{cases}$$

Note that, the $c^{m,n}_{u,v}$ above may not describe a coupling structure. Nevertheless we recover the transition probability θ_i, from it by averaging the right marginals.

The "Averaged Expectations" Heuristic. In contrast to the previous case, the second heuristic will update θ_i by directly averaging the expected values of $Z^{m,n}_{u,v}$ as follows

$$\theta_i(n)(v) = \begin{cases} \theta_{i-1}(n)(v) & \text{if } \exists m \in M.\, n\ R_{\mathcal{C}}\ m \\ \dfrac{\sum_{m,u \in M} \mathbf{E}[Z^{m,n}_{u,v} | \mathcal{C}]}{\sum_{x \in N} \sum_{m,u \in M} \mathbf{E}[Z^{m,n}_{u,x} | \mathcal{C}]} & \text{otherwise}\,. \end{cases}$$

Computing the Expected Values. We compute $\mathbf{E}[Z^{m,n}_{u,v} | \mathcal{C}]$ using a variant of the *forward-backward* algorithm for hidden Markov models. Let $Z^{m,n} \colon (M \times N)^\omega \to \mathbb{N}$ be the random variable that counts the number of occurrences of the pair (m,n) in a prefix in $\cong^*(R_{\mathcal{C}} \cup \bot)$ of the path. We compute the expected value of $Z^{m,n}$ w.r.t. the probability induced by $\mathcal{C}_\lambda$ as the solution $z_{m,n}$ of the following system of equations

$$z_{m,n} = \begin{cases} 0 & \text{if } m \ncong n \\ \iota(m,n) + \lambda \sum_{u,v} (z_{u,v} + 1) \cdot \mathcal{C}(u,v)(m,n) & \text{otherwise}\,, \end{cases}$$

where ι denotes the characteristic function for $\{(m_0, n_0)\}$. Then, the expected value of $Z^{m,n}_{u,v}$ w.r.t. the probability induced by $\mathcal{C}_\lambda$ is given by $\mathbf{E}[Z^{m,n}_{u,v} | \mathcal{C}] = \lambda \cdot z_{m,n} \cdot \mathcal{C}(m,n)(u,v) \cdot \beta^{\mathcal{C}}_\lambda(u,v)$.

[2] By abusing the notation, whenever the nominator is 0, we consider entire expression equal to 0, regardless of any division by 0. The same convention is used implicitly in the rest of the section.

Table 1 Comparison of the performance of Algorithm 1 on the IPv4 zeroconf protocol and the classic Drunkard's Walk w.r.t. the heuristics AM and AE.

Case	$\|M\|$	k	$\lambda = 1$				$\lambda = 0.8$			
			δ_λ-init	δ_λ-final	h	time	δ_λ-init	δ_λ-final	h	time
IPv4 (AM)	53	5	0.856	0.062	3	25.7	0.667	0.029	3	25.9
	103	5	0.923	0.067	3	116.3	0.734	0.035	3	116.5
	53	6	0.757	0.030	3	39.4	0.544	0.011	3	39.4
	103	6	0.837	0.032	3	183.7	0.624	0.017	3	182.7
IPv4 (AE)	53	5	0.856	0.110	2	14.2	0.667	0.049	3	21.8
	103	5	0.923	0.110	2	67.1	0.734	0.049	3	100.4
	53	6	0.757	0.072	2	21.8	0.544	0.019	3	33.0
	103	6	0.837	0.072	2	105.9	0.624	0.019	3	159.5
DrkW (AM)	39	7	0.565	0.466	14	259.3	0.432	0.323	14	252.8
	49	7	0.568	0.460	14	453.7	0.433	0.322	14	420.5
	59	8	0.646	–	–	TO	0.423	–	–	TO
DrkW (AE)	39	7	0.565	0.435	11	156.6	0.432	0.321	2	28.6
	49	7	0.568	0.434	10	247.7	0.433	0.316	2	46.2
	59	8	0.646	0.435	10	588.9	0.423	0.309	2	115.7

Choosing the initial approximant. Similarly to EM algorithms, the choice of the initial approximant $\mathcal{N}_0$ may have a significant effect on the quality of the solution. For the labeling of the states, one should follow Lemma 7. As for the choice of the underlying structure one shall be guided by Lemma 15. However, due to Theorem 14, it seems unlikely to have generic good strategies for a starting approximant candidate. Nevertheless, good selections for the transition probabilities may be suggested by looking at the problem instance.

Experimental Results Table 1 shows the results of some tests[3] on Algorithm 1. run on a number of instances $\langle \mathcal{M}, k \rangle$ of increasing size, where $\mathcal{M}$ is the bisimilarity quotient of either the IPv4 protocol [5, Ex.10.5] or the drunkard's walk, parametric on the number of states $|M|$. As initial approximant we use a suitably small instance of the same model. Each row reports the distance to the original model respectively from $\mathcal{N}_0$ and $\mathcal{N}_h$, where h is the total number of iterations; and execution time (in seconds). We compare the two heuristics, averaged marginals (AM) and averaged expectation (AE), on the same initial approximant.

The results obtained on the IPv4 protocol show significant improvements between the initial and the returned approximant. Notably, these are obtained in very few iterations. On this model, AM gives approximants of better quality compared with those obtained using AE; however AE seems to be slightly faster than AM. On the drunkard's walk model, the two heuristics exhibit opposite results w.r.t. the previous experiment: AE provides the best solutions with fewer iterations and significantly lower execution times.

7 Conclusions and Future Work

To the best of our knowledge, this is the first paper addressing the complexity of the *optimal* approximate minimization of MCs w.r.t. a behavioral metric semantics. Even though for a good evaluation of our heuristics more tests are needed, the current results seem promising. Moreover, in the light of [10, 3], relating the probabilistic bisimilarity distance to the LTL-model checking problem as $\delta_1(\mathcal{M}, \mathcal{N}) \geq |\mathbb{P}_{\mathcal{M}}(\varphi) - \mathbb{P}_{\mathcal{N}}(\varphi)|$, for all $\varphi \in$ LTL, our results might be used to lead saving in the overall model checking time. A deeper study of this topic

[3] The tests are done on a prototype implementation coded in Mathematica® (`http://people.cs.aau.dk/giovbacci/tools.html`) running on an Intel Core-i5 2.5GHz with 8GB of DDR3 RAM 1600MHz.

will be the focus of future work. We close with an interesting open problem. Membership of BA-λ in NP is left open. However, by arguments analogous to [11, 14] and leveraging on the ideas that made us produce the MC in Example 10, we suspect that BA-λ is hard for the square-root-sum problem. The latter is known to be NP-hard and in PSPACE, but membership in NP has been open since 1976. Allender et al. [1] showed that it can be decided in the 4th level of the counting hierarchy, thus it is unlikely its PSPACE-completeness.

References

1 Eric Allender, Peter B urgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009. doi:10.1137/070697926.

2 Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, David L. Dill, and Howard Wong-Toi. Minimization of timed transition systems. In *CONCUR*, volume 630 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 1992. doi:10.1007/BFb0084802.

3 Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. Converging from Branching to Linear Metrics on Markov Chains. In *ICTAC*, volume 9399 of *LNCS*, pages 349–367. Springer, 2015. doi:10.1007/978-3-319-25150-9_21.

4 Christel Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 1996. doi:10.1007/3-540-61474-5_57.

5 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

6 Borja Balle, Prakash Panangaden, and Doina Precup. A canonical form for weighted automata and applications to approximate minimization. In *LICS*, pages 701–712. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.70.

7 Michael Benedikt, Rastislav Lenhardt, and James Worrell. LTL Model Checking of Interval Markov Chains. In *TACAS*, volume 7795 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2013. doi:10.1007/978-3-642-36742-7_3.

8 Stefan Blom and Simona Orzan. A distributed algorithm for strong bisimulation reduction of state spaces. *International Journal on Software Tools for Technology Transfer*, 7(1):74–86, 2005. doi:10.1007/s10009-004-0159-4.

9 John F. Canny. Some Algebraic and Geometric Computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88)*, pages 460–467. ACM, 1988. doi:10.1145/62212.62257.

10 Di Chen, Franck van Breugel, and James Worrell. On the Complexity of Computing Probabilistic Bisimilarity. In *FoSSaCS*, volume 7213 of *LNCS*, pages 437–451. Springer, 2012. doi:10.1007/978-3-642-28729-9_29.

11 Taolue Chen and Stefan Kiefer. On the Total Variation Distance of Labelled Markov Chains. In *CSL-LICS'14*, pages 33:1–33:10. ACM, 2014. doi:10.1145/2603088.2603099.

12 Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for Labeled Markov Systems. In *CONCUR*, volume 1664 of *LNCS*, pages 258–273. Springer, 1999. doi:10.1007/3-540-48320-9_19.

13 Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled Markov processes. *Theoretical Compututer Science*, 318(3):323–354, 2004. doi:10.1016/j.tcs.2003.09.013.

14 Kousha Etessami and Mihalis Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1):1:1–1:66, 2009. doi:10.1145/1462153.1462154.

15 Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov Decision Processes. In *UAI*, pages 162–169. AUAI Press, 2004.

16 Giuliana Franceschinis and Richard R. Muntz. Bounds for quasi-lumpable markov chains. *Perform. Eval.*, 20(1-3):223–243, 1994. doi:10.1016/0166-5316(94)90015-9.

17 John Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971. doi:10.1016/B978-0-12-417750-5.50022-1.

18 Chi-Chang Jou and Scott A.Smolka. Equivalences, congruences, and complete axiomatizations for probabilistic processes. In *CONCUR'90 Theories of Concurrency: Unification and Extension*, volume 458 of *LNCS*, pages 367–383, 1990. doi:10.1007/BFb0039071.

19 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In *Proceedings of the 2nd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 228–240. ACM, 1983. doi:10.1145/800221.806724.

20 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990. doi:10.1016/0890-5401(90)90025-D.

21 Michal Ko1cmcvara and Michael Stingl. PENBMI 2.0. http://www.penopt.com/penbmi.html. Accessed: 2016-08-28.

22 Michal Ko1cmcvara and Michael Stingl. PENNON: A code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 18(3):317–333, 2003. doi:10.1080/1055678031000098773.

23 Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

24 David Lee and Mihalis Yannakakis. Online minimization of transition systems (extended abstract). In *Annual ACM Symposium on Theory of Computing*, pages 264–274. ACM, 1992. doi:10.1145/129712.129738.

25 Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 2 edition, 2008.

26 Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980. doi:10.1007/3-540-10235-3.

27 Edward F. Moore. Gedanken Experiments on Sequential Machines. In *Automata Studies*, pages 129–153. Princeton University, 1956.

28 Franck van Breugel and James Worrell. Towards Quantitative Verification of Probabilistic Transition Systems. In *ICALP*, volume 2076 of *LNCS*, pages 421–432, 2001.

29 Franck van Breugel and James Worrell. Approximating and computing behavioural distances in probabilistic transition systems. *Theoretical Computer Science*, 360(3):373–385, 2006. doi:10.1016/j.tcs.2006.05.021.

30 Mihali Yannakakis and David Lee. An efficient algorithm for minimizing real-time transition systems. *Formal Methods in System Design*, 11(2):113–136, 1997. doi:10.1023/A:1008621829508.

31 Shipei Zhang and Scott A. Smolka. Towards efficient parallelization of equivalence checking algorithms. In *FORTE*, volume C-10 of *IFIP Transactions*, pages 121–135. North-Holland, 1992.

Expressiveness of Probabilistic Modal Logics

Nathanaël Fijalkow[*1], **Bartek Klin**[†2], **and Prakash Panangaden**[‡3]

1 **University of Warwick, Warwick, UK**
nfijalkow@turing.ac.uk
2 **University of Warsaw, Warsaw, Poland**
klin@mimuw.edu.pl
3 **McGill University, Montreal, Canada**
prakash@cs.mcgill.ca

Abstract

Labelled Markov processes are probabilistic versions of labelled transition systems. In general, the state space of a labelled Markov process may be a continuum. Logical characterizations of probabilistic bisimulation and simulation were given by Desharnais et al. These results hold for systems defined on *analytic* state spaces and assume that there are countably many labels in the case of bisimulation and finitely many labels in the case of simulation.

In this paper, we first revisit these results by giving simpler and more streamlined proofs. In particular, our proof for simulation has the same structure as the one for bisimulation, relying on a new result of a topological nature. This departs from the known proof for this result, which uses domain theory techniques and falls out of a theory of approximation of Labelled Markov processes.

Both our proofs assume the presence of countably many labels. We investigate the necessity of this assumption, and show that the logical characterization of bisimulation may fail when there are uncountably many labels. However, with a stronger assumption on the transition functions (continuity instead of just measurability), we can regain the logical characterization result, for arbitrarily many labels. These new results arose from a new game-theoretic way of understanding probabilistic simulation and bisimulation.

1998 ACM Subject Classification F.1.2 [Modes of Computation] Probabilistic Computation, F.4.1 Mathematical Logic

Keywords and phrases probabilistic modal logic, probabilistic bisimulation, probabilistic simulation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.105

1 Introduction

It is now 40 years since the logical characterization of bisimulation was established by van Benthem [14] and by Hennessy and Milner [10] for nondeterministic transition systems. It was shown that two states (or processes) are bisimilar if and only if they satisfied the same formulas of a modal logic that has come to be called Hennessy-Milner logic. The probabilistic version was studied by Larsen and Skou [12] who defined probabilistic bisimulation for discrete probabilistic transition systems and established a logical characterization theorem for discrete systems with a strong finite-branching assumption.

* Supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1.
† Supported by the Polish National Science Centre (NCN) grant 2012/07/E/ST6/03026.
‡ Supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 105; pp. 105:1–105:12

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

The analysis of bisimulation was extended to continuous state spaces by Blute et al. [4] and a logical characterization was shown by Desharnais et al. [6, 7] who proved the result without using any negative constructs in the logic nor any kind of finite branching assumptions. Their proofs marked a departure from the usual combinatorial arguments and used some non-trivial results from measure theory, specifically using remarkable properties of analytic spaces; see [13] for an expository account.

The fact that the logical characterization result can be established with a purely positive logic was a surprise at the time. It opened the door to the possibility that there could be a logical characterization of *simulation*; we define this precisely below but the idea should be intuitively clear. A clever example, due to Josée Desharnais [8], showed that this cannot be done with the same logic as the one used for bisimulation; one needs to have disjunction in the logic. A logical characterization of simulation was proved [8] for transition systems with *finitely many* labels. The main contribution of [8] was approximation theory which included a domain-theoretic treatment; the logical characterization result fell out of the domain theory results. Desharnais [5] in her thesis gave a proof not using domain theory in the discrete case. What remained unknown until now is a proof that works for countably many labels, continuous state spaces and, if possible, not using domain theory. We provide such a result, extending the characterization for simulation to countably many labels with a proof very much analogous to the one given for bisimulation.

1.1 Results

1. We give a characterization of bisimulation and simulation in terms of Spoiler/Duplicator games. This is elementary but interesting: it was the main driver of the intuitions that led to the proofs of the present paper though, in the end, one does not actually need the games to establish the results.
2. The logical characterization of bisimulation has a proof which has a structure which can be boiled down to two main ingredients: Dynkin's π-λ theorem and the Unique Structure Theorem for analytic spaces. For simulation, it turns out that a completely analogous proof exists. It is enough to replace the two main ingredients by new positive versions: a positive analogue of the monotone class theorem and a positive UST, both of which we prove. This simplifies the previous domain-theoretic proof and clarifies the picture. The small price to pay is to move from analytic spaces to Polish ones; moving back to analytic is future work.
3. Both proofs rely on the countability of the set of formulas. This is necessary, as an explicit counterexample shows. But if the transition structure is continuous, logical characterization results are regained for arbitrary sized sets of labels. As far as we know, this is the first result of this type for uncountable label spaces.

Both logical characterization proofs, for bisimulation and simulation, have a similar structure and can even be said to follow the same *top-level* strategy as the original Hennessy-Milner proof.

2 Probabilistic systems and logics

We review some definitions and concepts from measure theory and topology. We assume that the reader is familiar with concepts like: σ-algebra, measurable functions, measures, topology and continuity.

Given a topological space X the σ-algebra induced by its open sets (or its closed sets) is called the *Borel algebra*; we will always work with Borel algebras of topological spaces. We call them Borel spaces.

A topological space is said to be *separable* if it has a countable dense subset. For metric spaces this is equivalent to having a countable base of open sets. A Polish space is the topological space underlying a complete separable metric space. Note that a space like $(0,1)$ which is not complete in its usual metric is nevertheless Polish, since it can be given a complete metric that produces the same topology. If X, Y are Polish spaces and $f : X \to Y$ is a continuous function then the image $f(X) \subset Y$ is an *analytic* space. The class of analytic spaces is not altered if we allow f to be measurable instead of continuous or if we take the image of a Borel set instead of all of X.

▶ **Definition 1.** A **Markov kernel** on a Borel space (X, Σ) is a function $\tau : X \times \Sigma \to [0,1]$ such that for each fixed $x \in X$, the set function $\tau(x, \cdot)$ is a sub-probability measure, and for each fixed $C \in \Sigma$ the function $\tau(\cdot, C)$ is a measurable function.

One interprets $\tau(x, C)$ as the probability of the process starting in state x making a transition into one of the states in C.

▶ **Definition 2.** A **labelled Markov process** (LMP) $\mathcal{S}$ with label set $\mathcal{A}$ is a structure $(X, \Sigma, \{\tau_a \mid a \in \mathcal{A}\})$, where (X, Σ) is a Borel space and

$$\tau_a : X \times \Sigma \longrightarrow [0,1]$$

is a Markov kernel for each $a \in \mathcal{A}$.

A key concept is bisimulation. The following definition is adapted from Larsen and Skou [12] to deal with measurability issues.

▶ **Definition 3** (Bisimulation)**.** Let $\mathcal{S} = (X, \Sigma, \tau)$ be a labelled Markov process. An equivalence relation R on X is a **bisimulation** if whenever xRy, with $x, y \in X$, we have that for all $a \in \mathcal{A}$ and every R-closed measurable set $C \in \Sigma$, $\tau_a(x, C) = \tau_a(y, C)$. We say that x and y are bisimilar, denoted $x \approx y$, if there exists a bisimulation R such that xRy.

The modal logic $\mathcal{L}_\wedge$ used in the logical characterization theorem of [7] is generated by the grammar:

$$\phi ::= \top \mid \phi \wedge \phi \mid \langle a \rangle_p \phi$$

where p ranges over rational numbers between 0 and 1. A state x satisfies the modal formula $\langle a \rangle_p \phi$ if there exists a measurable subset C with every state in C satisfying ϕ and $\tau_a(x, C) > p$. It is easy to show that the sets defined by formulas $[\![\phi]\!] := \{x \mid x \models \phi\}$ are all measurable. We write $x \equiv_\wedge y$ to say that x and y satisfy the same formulas in $\mathcal{L}_\wedge$.

The logical characterization theorem for probabilistic bisimulation is:

▶ **Theorem 4** ([7])**.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is analytic and $\mathcal{A}$ is countable, and for any $x, y \in X$, we have that $x \equiv_\wedge y$ if and only if $x \approx y$.* ◀

For a preorder R on a set X, we say that $C \subseteq X$ is *R-closed* if $x \in C$ and xRy implies $y \in C$, for all $x, y \in X$.

▶ **Definition 5** (Simulation)**.** Let $\mathcal{S} = (X, \Sigma, \tau)$ be a labelled Markov process. An preorder relation R on X is a **simulation** if whenever xRy, with $x, y \in X$, we have that for all $a \in \mathcal{A}$ and every R-closed measurable set $C \in \Sigma$, $\tau_a(x, C) \leq \tau_a(y, C)$. We say that x is simulated by y, denoted $x \lesssim y$, if there exists a simulation R such that xRy.

The logic $\mathcal{L}_{\vee\wedge}$ extends $\mathcal{L}_{\wedge}$ with disjunction:

$$\phi \;=\; \top \;\mid\; \phi \wedge \phi \;\mid\; \phi \vee \phi \;\mid\; \langle a \rangle_p \phi.$$

We write $x \leq_{\vee\wedge} y$ to say that every formula in $\mathcal{L}_{\vee\wedge}$ satisfied by x is also satisfied by y.

The previous logical characterization theorem for probabilistic simulation is:

▶ **Theorem 6** ([8]). *For any labelled Markov process (X, Σ, τ) where (X, Σ) is analytic and $\mathcal{A}$ is finite, and for any $x, y \in X$, we have that $x \leq_{\vee\wedge} y$ if and only if $x \lesssim y$.*

Existing proofs of Theorems 4 and 6 span several pages each, and are markedly dissimilar. In particular, the latter relies on the machinery of domain theory. One of our main contributions is to provide new, short proofs of both results.

3 Probabilistic (bi)simulation games

The classical notion of bisimulation and simulation for nondeterministic processes has a simple and elegant characterization in terms of games. These games, played between two players named Spoiler (who tries to prove that some two states in a process are not bisimilar) and Duplicator (who claims the opposite), provide convenient intuitions about the essence of bisimilarity.

To the best of our knowledge, probabilistic bisimulation and simulations have not been characterized by games before. In this section we fill this gap; as we shall see, the relevant games have a pleasantly simple structure, even in the setting of continuous space processes.

We begin with the case of **bisimulation game**. As in the classical case, we consider a spoiler/duplicator game with two players. Duplicator's plays are pairs of states that she claims are bisimilar. Spoiler tries to show that a given pair of states proffered by Duplicator are not bisimilar. Let $\mathcal{S} = (X, \Sigma, \tau)$ be a labelled Markov process, and $x, y \in X$. The bisimulation game starting from the position (x, y) alternates between moves of the following kinds:

- Spoiler chooses $a \in A$ and $C \in \Sigma$ such that $\tau_a(x, C) \neq \tau_a(y, C)$,
- Duplicator answers by choosing $x' \in C$ and $y' \notin C$ and the game continues from (x', y').

A player who cannot make a move at any point loses. Duplicator wins if the game goes on forever.

Note that the only way for Spoiler to win is to choose $C = X$ at some point; otherwise Duplicator can always choose some x' and y' as prescribed. (The only other situation where Duplicator cannot proceed would be $C = \emptyset$, but that is not a legal move for Spoiler since always $\tau_a(x, \emptyset) = \tau_a(y, \emptyset) = 0$.) On the other hand, Duplicator can win either by forcing an infinite play or by reaching a position (x, y) where $\tau_a(x, C) = \tau_a(y, C)$ for every $C \in \Sigma$.

The intuition behind the game should be clear. Spoiler tries to prove that states x and y are not bisimilar by showing a label a and a set C, purportedly closed under bisimilarity, such that the probabilities of a-labelled transitions to C are different for x and y. This difference of probabilities is checked but not disputed by Duplicator, who instead claims that C, in fact, is not closed under bisimilarity. She does that by choosing $x' \in C$ and $y' \notin C$ and claiming that these two are bisimilar; the game then proceeds in the same fashion.

Before we formally prove the correctness of this game, let us spend a moment to consider what makes a game-theoretic characterization "elegant". In our opinion, the classical bisimulation game for nondeterministic processes is elegant because it allows one to characterize a global property of behaviours (bisimilarity) in terms of a game whose rules only depend on local considerations. Indeed, whether a move in the game is legal or not does not depend on

bisimilarity or other long-range properties, but merely on local observations about transition capabilities that cannot be disputed by either player.

We argue that this criterion of elegance is satisfied by our probabilistic game. One can imagine the players engaging in a brief experiment with the given Markov process after each move by Spoiler, to determine that the two transition probabilities involved are indeed different. By performing random a-transitions from x and y sufficiently many times, Spoiler can demonstrate to Duplicator, with an arbitrarily high confidence level, that the probabilities of getting to C are different and so that the move to C is legal for Spoiler. It is important to note, comparing the game to the definition of probabilistic bisimulation itself, that the legality of a Spoiler's move does not depend on the set C being actually closed under bisimilarity; a game with such a condition would not be "elegant".

The question of *how many* random transitions are enough to convince Duplicator that a Spoiler's move is legal, and hence how much time it takes for Spoiler to win the game if x and y are not bisimilar, suggests a potentially interesting connection of the bisimulation game to the quantitative framework of metrics on labelled Markov processes [9]. We leave this for future work.

Back to formal development. Since all infinite plays are won by the same player (Duplicator), standard game-theoretic arguments prove that:

▶ **Fact 7.** *The bisimulation game is determined, i.e., from every position (x, y) either Spoiler has a winning strategy or Duplicator does.* ◀

From this we infer:

▶ **Theorem 8.** *The states x and y are bisimilar if and only if Duplicator has a winning strategy from (x, y).*

Proof. For the left-to-right implication, for bisimilar x and y, we construct a winning strategy from (x, y) for Duplicator. In this strategy, for any move a and C by Spoiler, Duplicator chooses some arbitrary $x' \in C$ and $y' \notin C$ such that x' and y' are bisimilar. This is always possible: since Spoiler's move was legal, and it originated from a pair of bisimilar states, C cannot be closed under bisimilarity. This strategy is winning for Duplicator since it allows her response to any move by Spoiler, and Duplicator wins all infinite plays.

For the right-to-left implication, we shall show that the set R of all pairs (x, y) whence Duplicator has a winning strategy, is a bisimulation. To this end, first we need to show that R is an equivalence relation. Reflexivity is immediate, since from a position (x, x) Spoiler has no legal moves. For symmetry, given a winning strategy from (x, y) Duplicator builds a strategy from (y, x) trivially: she simply replies to any first move by Spoiler as if she would reply to a move from (x, y), and then she follows the given strategy. For transitivity, assume winning strategies for Duplicator from (x, y) and (y, z). A winning strategy for (x, z) works as follows: for a legal move a and C from Spoiler, it must be that $\tau_a(x, C) \neq \tau_a(y, C)$ or $\tau_a(y, C) \neq \tau_a(z, C)$. Depending on which of these inequalities holds, reply according to the strategy from (x, y) or from (y, z), and then follow that winning strategy.

Now assume towards contradiction that R is not a bisimulation. This means that for some x, y such that xRy, there exists a letter a in A and an R-closed subset C of X such that $\tau_a(x, C) \neq \tau_a(y, C)$. Consider a and C as a Spoiler's move from (x, y). No matter what Duplicator chooses as $x' \in C$ and $y' \notin C$, since C is R-closed we have that not $(x'Ry')$ and, by Fact 7, Spoiler has a winning strategy from (x', y'). This forms a winning strategy for Spoiler from (x, y), contradicting the assumption that xRy. ◀

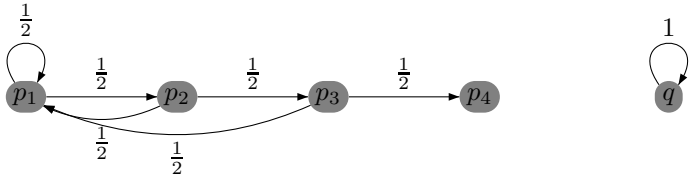


Figure 1 It takes four steps for Spoiler to convince Duplicator that the state p_1 does not simulate q.

Simulation game is defined in a very similar fashion, alternating the following moves:

- Spoiler chooses $a \in A$ and $C \in \Sigma$ such that $\tau_a(x, C) > \tau_a(y, C)$,
- Duplicator answers by choosing $x' \in C$ and $y' \notin C$ and the game continues from (x', y').

Again, a player who cannot make a move at any point loses, and Duplicator wins all infinite plays.

The intuition behind the game is as before, except now Spoiler maintains that his chosen sets C are $\lesssim$-closed, and Duplicator contradicts that by choosing $x' \in C$ and $y' \notin C$ and maintaining that $x' \lesssim y'$. All other considerations remain virtually the same, up to and including:

▶ **Theorem 9.** *$x \lesssim y$ if and only if Duplicator has a winning strategy from (x, y).*

▶ **Example 10.** We illustrate the simulation game on an example (see Fig. 1). In this Markov process there is only one label. From the state q, the process loops forever. On the other hand, from the state p_1, one can reach the deadlock state p_4 through the path to p_2 and p_3.

We examine the simulation game and how Spoiler can successfully prove to Duplicator that the state p_1 does not simulate q. We start the simulation game from (q, p_1). A possible first move is $C = \{q, p_2\}$ since $\tau(q, C) = 1 > \tau(p_1, C) = \frac{1}{2}$, but it allows Duplicator to play (q, p_1), back to the original position. A smarter move is $C = \{q, p_1\}$, to which Duplicator has several possible answers, all losing. For instance, if Duplicator plays (q, p_4), Spoiler wins immediately by choosing $C = X$. Duplicator may survive more steps by playing (q, p_2), then (q, p_3), before the fatal (q, p_4).

4 Logical characterization of bisimulation, revisited

In this section, we give a short proof for the logical characterization of bisimulation, which relies on two ingredients: the π-λ theorem and the Unique Structure Theorem.

4.1 The π-λ Theorem and the Unique Structure Theorem

A *π-system* is a family of subsets of a set X closed under finite intersections. A *λ-system* is a family that contains X and is closed under complement and countable disjoint unions. A *σ-algebra* is a family closed under complement, countable unions and countable intersections. For a family $\mathcal{E}$, let $\sigma(\mathcal{E})$ denote the least σ-algebra that contains $\mathcal{E}$.

▶ **Theorem 11** (Dynkin's π-λ theorem, [3]). *For any π-system Π and a λ-system Λ on the same set X, if $\Pi \subseteq \Lambda$ then $\sigma(\Pi) \subseteq \Lambda$.*

Below, $\equiv_{\mathcal{E}}$ is the relation of equivalence up to $\mathcal{E}$, i.e., $x \equiv_{\mathcal{E}} y$ if and only if, for every $Y \in \mathcal{E}$, $x \in Y$ iff $y \in Y$.

▶ **Theorem 12** (Unique Structure Theorem, [1]). *In any analytic space (X, Σ), for every countable family $\mathcal{E} \subseteq \Sigma$ such that $X \in \mathcal{E}$, every measurable, $\equiv_{\mathcal{E}}$-closed subset of X is an element of $\sigma(\mathcal{E})$.*

4.2 Logical Characterization

▶ **Theorem 13.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is analytic and $\mathcal{A}$ is countable, $\equiv_\wedge$ is a bisimulation.*

Proof. Take some $x, y \in X$ and assume that there exists some $a \in \mathcal{A}$ such that $\tau_a(x, C) \neq \tau_a(y, C)$ for some $\equiv_\wedge$-closed set $C \in \Sigma$. We need to prove that $x \not\equiv_\wedge y$.

Denote $\delta = \tau_a(x, -)$ and $\gamma = \tau_a(y, -)$. If $\delta(X) > \gamma(X)$, then pick a rational number p such that $\delta(X) > p > \gamma(X)$; it is easy so see that $x \models \langle a \rangle_p \top$ and $y \not\models \langle a \rangle_p \top$, therefore $x \not\equiv_\wedge y$. The same formula distinguishes x and y if $\delta(X) < \gamma(X)$.

If $\delta(X) = \gamma(X)$ then pick any $\equiv_\wedge$-closed $C \in \Sigma$ such that $\delta(C) \neq \gamma(C)$. Define

$$\Pi = \{[\![\phi]\!] \mid \phi \in \mathcal{L}_\wedge\} \qquad \text{and} \qquad \Lambda = \{Y \in \Sigma \mid \delta(Y) = \gamma(Y)\}.$$

It is easy to see that Π is a π-system and Λ is a λ-system (in particular, Λ is closed under complement since $\delta(X) = \gamma(X)$). Clearly, $\equiv_\Pi = \equiv_\wedge$. Moreover, since there are only countably many formulas, Π is countable and, by Theorem 12, $C \in \sigma(\Pi)$. Since by assumption $C \notin \Lambda$, we have $\sigma(\Pi) \not\subseteq \Lambda$, hence (by Theorem 11) $\Pi \not\subseteq \Lambda$. In other words, there exists an $\mathcal{L}_\wedge$ formula ϕ such that $\delta([\![\phi]\!]) \neq \gamma([\![\phi]\!])$.

Without loss of generality, assume $\delta([\![\phi]\!]) > \gamma([\![\phi]\!])$ and pick $p \in \mathbb{Q}$ such that $\delta([\![\phi]\!]) > p > \gamma([\![\phi]\!])$. We readily obtain $x \models \langle a \rangle_p \phi$ and $y \not\models \langle a \rangle_p \phi$, hence $x \not\equiv_\wedge y$ as requested. ◀

This easily implies Theorem 4, repeated here:

▶ **Corollary 14.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is analytic and $\mathcal{A}$ is countable, and for any $x, y \in X$, we have that $x \equiv_\wedge y$ if and only if $x \approx y$.*

Proof. The right-to-left implication is an easy induction on the structure of formulas. The left-to-right implication is immediate by Theorem 13. ◀

5 Logical characterization of simulation, revisited

Our proof of the logical characterization of simulation is completely analogous to the one for bisimulation. It is enough to replace the two main ingredients (Theorems 11 and 12) by new ones.

5.1 The Positive Monotone Class Theorem and the Positive Unique Structure Theorem

A *lattice of sets* is a family of subsets of a set X closed under finite unions and intersections.[1] A *monotone class* is a family closed under unions of increasing chains and under intersections of decreasing chains. A *σ-lattice of sets* is a family of sets closed under countable unions and countable intersections. For a family $\mathcal{E}$, let $L(\mathcal{E})$ denote the least σ-lattice of sets that contains $\mathcal{E}$.

[1] A lattice of sets is sometimes called *ring of sets*. However, in measure theory ring of sets means something else (a family closed under union and set difference), so we choose a different name.

▶ **Theorem 15** (Positive Monotone Class Theorem). *For any lattice of sets $\mathcal{E}$ and any monotone class $\mathcal{M}$ on the same set X, if $\mathcal{E} \subseteq \mathcal{M}$ then $L(\mathcal{E}) \subseteq \mathcal{M}$.*

This result is similar and easier to prove than Theorem 11, and it should be treated as folklore.

Below, $\sqsubseteq_{\mathcal{E}}$ is the preorder determined by $\mathcal{E}$, i.e., $x \sqsubseteq_{\mathcal{E}} y$ if and only if, for every $Y \in \mathcal{E}$, $x \in Y$ implies $y \in Y$.

▶ **Theorem 16** (Positive Unique Structure Theorem). *In any Polish space (X, Σ), for every countable family $\mathcal{E} \subseteq \Sigma$, every nonempty, different from X, measurable and $\sqsubseteq_{\mathcal{E}}$-closed subset of X is an element of $L(\mathcal{E})$.*

This result strengthens Theorem 12, albeit on the restricted domain of Polish spaces. (Extending it to analytic spaces is future work.) Its proof is also more involved, using ideas similar to the proof of Lusin's Separation Theorem for analytic sets (see [11]). The proof was pointed out to us by Roman Pol.

5.2 The logical characterization

▶ **Theorem 17.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is Polish and $\mathcal{A}$ is countable, $\leq_{\wedge}$ is a simulation.*

Proof. Take some $x, y \in X$ and assume that there exists some $a \in A$ such that $\tau_a(x, C) > \tau_a(y, C)$ for some $\leq_{\wedge}$-closed set $C \in \Sigma$. We need to prove that $x \not\leq_{\wedge} y$.

Denote $\delta = \tau_a(x, -)$ and $\gamma = \tau_a(y, -)$. Pick any $\leq_{\wedge}$-closed $C \in \Sigma$ such that $\delta(C) > \gamma(C)$. Then C cannot be empty, since $\delta(\emptyset) = \gamma(\emptyset) = 0$. If $C = X$, pick a rational number p such that $\delta(X) > p > \gamma(X)$; it is easy so see that $x \models \langle a \rangle_p \top$ and $y \not\models \langle a \rangle_p \top$, therefore $x \not\leq_{\wedge} y$.

If $C \neq X$, define

$$\mathcal{E} = \{[\![\phi]\!] \mid \phi \in \mathcal{L}_{\wedge}\} \qquad \text{and} \qquad \mathcal{M} = \{Y \in \Sigma \mid \delta(Y) \leq \gamma(Y)\}.$$

It is easy to see that $\mathcal{E}$ is a lattice of sets and (by continuity of measure) $\mathcal{M}$ is a monotone class. Clearly, $\sqsubseteq_{\mathcal{E}} = \leq_{\wedge}$. Moreover, since there are only countably many formulas, $\mathcal{E}$ is countable hence, by Theorem 16, $C \in L(\mathcal{E})$. Since by assumption $C \notin \mathcal{M}$, we have $L(\mathcal{E}) \not\subseteq \mathcal{M}$, hence (by Theorem 15) $\mathcal{E} \not\subseteq \mathcal{M}$. In other words, there exists a formula ϕ such that $\delta([\![\phi]\!]) > \gamma([\![\phi]\!])$. Pick $p \in \mathbb{Q}$ such that $\delta([\![\phi]\!]) > p > \gamma([\![\phi]\!])$. We readily obtain $x \models \langle a \rangle_p \phi$ and $y \not\models \langle a \rangle_p \phi$, hence $x \not\leq_{\wedge} y$ as requested. ◀

Compared to Theorem 6, the following easy consequence is restricted to Polish spaces but generalized to countable sets of labels.

▶ **Corollary 18.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is Polish and $\mathcal{A}$ is countable, and for any $x, y \in X$, we have that $x \leq_{\wedge} y$ if and only if $x \lesssim y$.*

Proof. The right-to-left implication is an easy induction on the structure of formulas. The left-to-right implication is immediate by Theorem 17. ◀

6 The case of uncountably many labels

Our proofs of the logical characterizations for simulation and bisimulation rely on the assumption that the set of formulas (and, equivalently, the set of transition labels) is countable. In this section we investigate the necessity of this assumption. We first observe that indeed if there are uncountably many labels, then the logical characterization fails in general. However, we show that if the transition structure is continuous, then the logical characterization holds again, without any assumption on the set of labels.

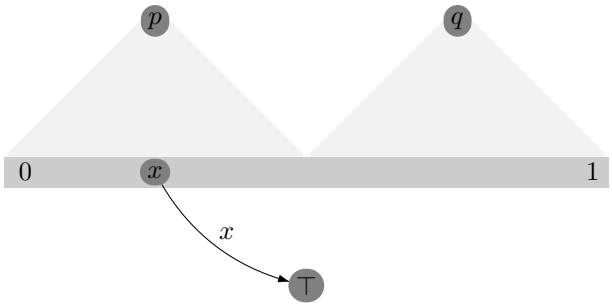


Figure 2 The two states p and q do not simulate each other, but they satisfy the same formulas of $\mathcal{L}_{\curlywedge}$.

6.1 A counterexample

In the classical logical characterization of (bi)similarity for nondeterministic labelled transition systems [10], one can restrict to a logic with finite conjunction and disjunction only if the systems in question satisfy a finite branching property called image finiteness: each state can have only finitely many successors for any given transition label. Since [6, 7] it has been known that this restriction does not apply to probabilistic systems, where a finitary logic is enough to characterize bisimilarity on systems with arbitrary (probabilistic) branching.

On the other hand, in the classical nondeterministic setting, once image finiteness is ensured, the size of the set of transition labels matters very little. Even if infinitely many, or even uncountably many labels are permitted, a finitary logic (with a correspondingly large set of modal operators) is enough to characterize (bi)similarity for nondeterministic transition systems labelled with them.

We now show that this is not the case for labelled Markov processes with continuous state spaces. Specifically, we show an example where the set of labels is uncountable and the logical characterization fails, even though the space of states is a particularly simple, compact Polish space.

Denote $X = \{p, q, \top\} \cup [0, 1]$. We equip X with the smallest σ-algebra that makes all Borel sets of $[0, 1]$ as well as the singletons $\{p\}, \{q\}$ and $\{\top\}$ measurable. Denote by μ the Lebesgue[2] probability measure on $[0, 1]$.

Consider a set of actions $\mathcal{A} = [0, 1]$. Define functions $\tau_a : X \times \Sigma \to [0, 1]$ for each $a \in \mathcal{A}$ as follows:

$$\begin{array}{rcl} \tau_a(p, C) & = & \mu(C \cap [0, \frac{1}{2}]) \\ \tau_a(q, C) & = & \mu(C \cap [\frac{1}{2}, 1]) \\ \tau_a(x, \top) & = & \begin{cases} 1 & \text{if } x = a \\ 0 & \text{otherwise} \end{cases} \end{array}$$

The following proposition easily implies that logical characterizations both for bisimulation and for simulation fail for this labelled Markov process.

▶ **Proposition 19.** *Neither p nor q simulates the other, but they satisfy the same formulas of $\mathcal{L}_{\curlywedge}$.*

[2] We mean the usual measure on $[0, 1]$ which assigns to intervals their length. However this is usually extended to the Lebesgue σ-algebra, i.e., the one obtained by completing the Borel σ-algebra with respect to this measure. We are just using this measure on the Borel sets.

Proof. We prove that neither p nor q simulates the other. First, for any x, y in $[0,1]$, if $x \neq y$ then neither of these simulates the other. Indeed, from x, the action $a = x$ leads to $\top$ with probability 1 and leads nowhere from y. It follows that every subset of $[0,1]$ is $\lesssim$-closed; in particular this applies to $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$. This implies that neither p nor q simulates the other, because $\tau_a(p, [0, \frac{1}{2}]) = 1$ and $\tau_a(q, [0, \frac{1}{2}]) = 0$, and vice-versa $\tau_a(p, [\frac{1}{2}, 1]) = 0$ and $\tau_a(q, [\frac{1}{2}, 1]) = 1$.

To see that p and q satisfy the same formulas, we observe that for every finite subset $\mathcal{B} \subseteq \mathcal{A}$, p and q do simulate each other (indeed, they are even bisimilar) in the system restricted to labels from $\mathcal{B}$. The claim easily follows from this, since every formula of $\mathcal{L}_{\vee\wedge}$ uses finitely many labels.

So for a finite $\mathcal{B} \subseteq \mathcal{A}$, define a relation R on X to be the least equivalence relation such that pRq and xRy for each $x, y \in [0,1] \setminus \mathcal{B}$. We claim that R is a bisimulation on the system restricted to labels with $\mathcal{B}$. The only nontrivial case is the pair pRq: every R-closed set $C \subseteq [0,1]$ is either finite or co-finite, from which it easily follows that $\tau_a(p, C) = \tau_a(q, C)$. ◀

Intuitively, the core of the problem here is the highly non-continuous nature of transitions from $[0,1]$, allowing one to observe specific states from that uncountable space. Indeed, as we show in the following section, the problem disappears and the logical characterizations hold if we assume that the transition function $\tau_a(\cdot, C)$ is continuous for each a and C.

6.2 Logical characterizations for continuous transition functions

Given a labelled Markov process (X, Σ, τ) with labels from a set $\mathcal{A}$, we denote by $(X, \Sigma, \tau_{|\mathcal{B}})$ the same system restricted to labels from $\mathcal{B} \subseteq \mathcal{A}$.

▶ **Theorem 20.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is Polish and such that for all $a \in \mathcal{A}, C \in \Sigma$, the function $\tau_a(\cdot, C)$ is continuous, there exists a countable set $\mathcal{B}$ such that the bisimilarity relation $\approx$ on $(X, \Sigma, \tau_{|\mathcal{B}})$ coincides with that on (X, Σ, τ).*

Proof. We will use the fact that, under the above assumptions, X^2 is also a Polish space for the product topology, hence it satisfies the *hereditary Lindelöf property*: any open cover of a subset of X^2 has a countable subcover.

By definition, the bisimilarity relation $\approx$ on (X, Σ, τ) is the largest bisimulation. It is standard to define it as the greatest fixpoint of a certain operator on binary relations on X. For us it will be convenient to speak in terms of complements, and we consider the *complement* of $\approx$ as the *least* fixpoint of the operator:

$$\Phi(R) = \left\{ (x,y) \in X^2 \;\middle|\; \exists a \in \mathcal{A}, \exists C \in \Sigma \ (X^2 \setminus R)\text{-closed, s.t. } \tau_a(x, C) \neq \tau_a(y, C) \right\}$$

Thanks to Tarski's fixed point theorem, this is obtained by defining a sequence $(W_\alpha)_\alpha$ of subsets of X^2 indexed by ordinals α: for $\alpha + 1$ a successor ordinal and β a limit ordinal, define:

$$\begin{array}{lcl} W_0 & = & \emptyset \\ W_{\alpha+1} & = & \left\{ (x,y) \in X^2 \;\middle|\; \exists a \in \mathcal{A}, \exists C \in \Sigma \ (X^2 \setminus W_\alpha)\text{-closed, s.t. } \tau_a(x, C) \neq \tau_a(y, C) \right\} \\ W_\beta & = & \bigcup_{\alpha < \beta} W_\alpha. \end{array}$$

The complement of $\approx$ on (X, Σ, τ) is the union of all W_α for all ordinals α. More specifically, $(W_\alpha)_\alpha$ form an increasing sequence that reaches a fixpoint at some ordinal γ not larger than the cardinality of $\mathcal{P}(X^2)$.

Note that all W_α are open sets in X^2. This is proved by ordinal induction: for a successor ordinal, $W_{\alpha+1}$ is a union of sets of the form

$$\{(x,y) \in X^2 \mid \tau_a(x,C) \neq \tau_a(y,C)\}$$

for some a and C. Such a set is open, since it is the inverse image of the (open) inequality relation on $[0,1]$ along the continuous function $\tau_a(\cdot, C)$.

For each ordinal α we construct a countable subset $\mathcal{B}_\alpha \subseteq \mathcal{A}$ such that W_α calculated on $(X, \Sigma, \tau_{|\mathcal{B}_\alpha})$ coincides with W_α calculated on (X, Σ, τ).

For sucessor ordinals, rewrite the definition of $W_{\alpha+1}$ as:

$$W_{\alpha+1} = \bigcup_{a \in \mathcal{A}} \{(x,y) \in X^2 \mid \exists C \in \Sigma\ (X^2 \setminus W_\alpha)\text{-closed, s.t. } \tau_a(x,C) \neq \tau_a(y,C)\}.$$

This is a union of open sets. Since X^2 is hereditary Lindelöf, one can extract a countable subcover of this union, indexed by some set $\mathcal{B} \subseteq \mathcal{A}$. It is then enough to take $\mathcal{B}_{\alpha+1} = \mathcal{B}_\alpha \cup \mathcal{B}$.

For limit ordinals, extract a countable subcover of the union $W_\beta = \bigcup_{\alpha<\beta} W_\alpha$ and take $\mathcal{B}_\beta$ to be the union of the $\mathcal{B}_\alpha$'s defined for α's from that subcover.

Now the countable set $\mathcal{B}_\gamma$, where γ is the ordinal for which W_γ reaches the least fixpoint of Φ, satisfies the desired property. ◀

The same result holds for simulation:

▶ **Theorem 21.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is Polish and such that for all $a \in \mathcal{A}, C \in \Sigma$, the function $\tau_a(\cdot, C)$ is continuous, there exists a countable set $\mathcal{B}$ such that the similarity preorder $\lesssim$ on $(X, \Sigma, \tau_{|\mathcal{B}})$ coincides with that on (X, Σ, τ).*

Proof. Completely analogous to the proof of Theorem 20, but with the operator

$$\Phi(R) = \{(x,y) \in X^2 \mid \exists a \in \mathcal{A}, \exists C \in \Sigma\ (X^2 \setminus R)\text{-closed, s.t. } \tau_a(x,C) > \tau_a(y,C)\}$$

instead. In particular the fact that each W_α is open, still holds. ◀

The following immediately follows from Theorems 20 and 21 in the light of Corollaries 14 and 18.

▶ **Corollary 22.** *For any labelled Markov process (X, Σ, τ) where (X, Σ) is Polish and such that for all $a \in \mathcal{A}, C \in \Sigma$, the function $\tau_a(\cdot, C)$ is continuous, for any $x, y \in X$,*

- *$x \equiv_\wedge y$ if and only if $x \approx y$,*
- *$x \leq_\curlywedge y$ if and only if $x \lesssim y$.*

7 Conclusions

The results of this paper suggest that we have arrived at a deeper understanding of the interplay of modal logic and probabilistic transition structure. Variations of the logic can also be used for logical characterization of bisimulation, for example, with the modal construct and just disjunction instead of just conjunction, as studied in [2]. The arguments are minor variations of the proofs given in Section 3sec:bisim. The earlier proof of logical characterization of simulation [8] emerged as a by-product of the theory of approximation; the proof of the present paper is direct. It is particularly pleasing that the two logical characterization proofs have the same general shape and also resemble the overall strategy of the Hennessy-Milner proof.

The game characterization, though elementary, is both pleasing and intriguing. As suggested earlier, there might be interesting links to metrics and the number of moves it takes for Spoiler to win a game. The connection between metrics and bisimulation is well understood but it is possible that via the game one might gain a more quantitative understanding of the numerical significance of the metric.

Acknowledgments. We are very much indebted to Roman Pol, who showed us the proof of Theorem 16 which had eluded us for a long time.

We would like to thank the Simons Institute for hosting the program *Logical Structures in Computation* during the Fall of 2016 where we were able to work together in a congenial atmosphere. We are grateful to Martin Otto and Thomas Colcombet for stimulating conversations in Berkeley.

References

1 W. Arveson. *An Invitation to C^*-Algebra*. Springer-Verlag, 1976.
2 M. Bernardo and M. Miculan. Disjunctive probabilistic modal logic is enough for bisimilarity on reactive probabilistic systems. In *ICTCS*, volume 1720, pages 203–220, 2016.
3 P. Billingsley. *Probability and Measure*. Wiley-Interscience, 1995.
4 R. Blute, J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labelled Markov processes. In *LICS*, 1997.
5 J. Desharnais. *Labelled Markov Processes*. PhD thesis, McGill University, 1999.
6 J. Desharnais, A. Edalat, and P. Panangaden. A logical characterization of bisimulation for labelled Markov processes. In *LICS*, pages 478–489, 1998.
7 J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labeled Markov processes. *Information and Computation*, 179(2):163–193, 2002.
8 J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Approximating labeled Markov processes. *Information and Computation*, 184(1):160–200, 2003.
9 J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. A metric for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, June 2004.
10 M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In *ICALP*, volume 85, pages 299–309, 1980.
11 A. S. Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.
12 K. G. Larsen and A. Skou. Bisimulation through probablistic testing. *Information and Computation*, 94:1–28, 1991.
13 P. Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009.
14 J. van Benthem. *Modal correspondence theory*. PhD thesis, University of Amsterdam, 1976.

Emptiness of Zero Automata Is Decidable*†

Mikołaj Bojańczyk[1], Hugo Gimbert[2], and Edon Kelmendi[2]

1 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
`bojan@mimuw.edu.pl`
2 **LaBRI, Université de Bordeaux, CNRS, Bordeaux, France**
`hugo.gimbert@labri.fr`
3 **LaBRI, Université de Bordeaux, CNRS, Bordeaux, France**
`edon.kelmendi@labri.fr`

Abstract

Zero automata are a probabilistic extension of parity automata on infinite trees. The satisfiability of a certain probabilistic variant of MSO, called TMSO + ZERO, reduces to the emptiness problem for zero automata. We introduce a variant of zero automata called nonzero automata. We prove that for every zero automaton there is an equivalent nonzero automaton of quadratic size and the emptiness problem of nonzero automata is decidable, with complexity co-NP. These results imply that TMSO + ZERO has decidable satisfiability.

1998 ACM Subject Classification F.4.3 Formal Languages, F.4.1 Mathematical Logic

Keywords and phrases tree automata, probabilistic automata, monadic second-order logic

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.106

1 Introduction

In this paper, we prove that emptiness is decidable for two classes of automata, namely *zero* and *nonzero* automata. Zero automata were introduced as a tool for recognizing models of a probabilistic extension of MSO on infinite trees [1]. Nonzero automata, introduced in this paper, are equivalent to zero automata, but have simpler semantics.

Both zero and nonzero automata are probabilistic extensions of parity automata on infinite trees. Here we focus on the case of binary trees. The automaton performs a random walk on the infinite binary input tree: when the automaton is in a state q on a node labelled with a, it selects non-deterministically a transition (q, a, r_0, r_1) and moves with equal probability $\frac{1}{2}$ either to the left node in state r_0 or to the right node in state r_1.

The set of branches of the infinite binary tree is equipped with the uniform probability measure, which is used to define the acceptance condition. There are two variants of the acceptance condition, one for zero automata and one for nonzero automata

A *nonzero* automaton is equipped with a total order $\leq$ on its set of states Q and three accepting subsets of states $F_\forall, F_1$ and $F_{>0}$. A run is accepting if:

(a) on every branch the limsup state (i.e. the maximal state seen infinitely often) is in $F_\forall$;
(b) and with probability 1 the limsup state is in F_1;
(c) and every time the run visits a state in $F_{>0}$ there is a nonzero probability that all subsequent states are in $F_{>0}$.

* Full version with proof is [2], `http://arxiv.org/abs/1702.06858`.

† The research of M. Bojańczyk is supported by the ERC grant LIPA under the Horizon 2020 framework. H. Gimbert and E. Kelmendi are supported by the French ANR project "Stoch-MC" and "LaBEX CPU" of Université de Bordeaux.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 106; pp. 106:1–106:13

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

Condition (a) is the classical parity condition for tree automata and condition (b) is equivalent to the qualitative condition from [5]. Condition (c) seems to be new. Conditions (a) and (b) are used to define the acceptance condition of zero automata as well, the difference between zero and nonzero automata lies in condition (c).

The paper [1] introduced a variant of MSO on infinite trees with a probabilistic quantifier, called TMSO+zero, inspired by probabilistic MSO from [9]. In the case where zero is the unary predicate which checks whether a set of branches has probability 0, the contribution of [1] was a proof that for every formula of this logic one can compute a zero automaton which accepts the same trees. The logic is powerful enough to formulate properties like "every node in the tree has a descendant node labelled with b and the set of branches with infinitely many b has probability 0". As argued in [1], the motivation for this logic is twofold. First, it extends various probabilistic logics known in the literature, e.g. qualitative probabilistic CTL* [8], or qualitative probabilistic CTL* extended with ω-regular path properties [3]. Second, the logic, although less general than MSO, represents a robust class of languages of infinite trees that goes beyond classical MSO, and thus falls under the scope of the programme of searching for decidable extensions of MSO.

The emptiness problem for zero automata was not solved in [1], thus leaving open the logic's decidability. A step toward an emptiness algorithm was made in [10], where it was shown that for subzero automata – the special case of zero automata where only conditions (a) and (b) are used – one can decide if the recognized language contains a regular tree. In this paper we prove that zero and nonzero automata have decidable emptiness, and therefore also the logic from [1] has decidable satisfiability.

The main results of this paper are:

(i) For every zero automaton there is an equivalent nonzero automaton of quadratic size.
(ii) A nonzero automaton with $F_\forall = Q$ is nonempty if and only if its language contains a regular tree of size $|Q|$. This is decidable in NP.
(iii) The emptiness problem of nonzero automata is in co-NP.

To prove (iii) we provide a reduction of the emptiness problem to the computation of the winner of a parity game called the *jumping game*. For that we rely on (ii): the states of the jumping game are regular runs of a nonzero automaton where $F_\forall = Q$. According to (i) the emptiness problem for zero automata is in co-NP as well.

These results were recently improved: the emptiness problem is actually in NP$\cap$co-NP, and even in PTIME if $F_\forall = Q$, see [2].

The plan of the paper is as follows. In Section 2 we introduce zero and nonzero automata and state our main result (iii) (Theorem 3). In Section 3 we show (i) (Lemma 5). In Section 4 we focus on the special case where $Q = F_\forall$ and show (ii) (Theorem 10). In Section 5 we introduce jumping games and combine the previous results to provide a proof of (iii).

2 Zero and nonzero automata

This section introduces trees and nonzero and zero automata.

Trees, branches and subtrees. The automata of this paper describe properties of infinite binary labelled trees. A node in a tree is a sequence in $\{0,1\}^*$. A *tree* over an alphabet Σ is a function $t : \{0,1\}^* \to \Sigma$. We use standard terminology for trees: node, root, left child, right child, leaf, ancestor and descendant. A *branch* is a sequence in $\{0,1\}^\omega$, viewed as an infinite sequence of left or right turns. A branch *visits* a node if the node is a prefix of the branch.

A *subtree* is a non-empty and ancestor-closed set of nodes. A subtree is *leaf-free* if each of its nodes has at least one child in the subtree. A branch of a subtree is a branch which visits only nodes of the subtree.

Probability measure over branches. We use the *coin-flipping* measure on $\{0,1\}^\omega$: each bit is chosen independently at random, with 0 and 1 having equal probability, and every Borel subset of $\{0,1\}^\omega$ is measurable. The probability of a subtree is the probability of the set of branches of the subtree. The inner regularity of the coin-flipping measure (see e.g. [7, Theorem 17.10]) implies:

▶ **Lemma 1.** *The probability of a measurable set E is the supremum of the probabilities of the subtrees whose every branch belongs to E.*

Nonzero automata. Intuitively, a nonzero automaton is a nondeterministic parity tree automaton which has the extra ability to check whether the set of branches satisfying the parity condition has zero or nonzero probability.

▶ **Definition 2.** The syntax of a nonzero automaton is a tuple

$$\underbrace{Q}_{\text{states}} \qquad \underbrace{\Sigma}_{\text{input alphabet}} \qquad \underbrace{\Delta \subseteq Q \times \Sigma \times Q^2}_{\text{transitions}},$$

with all components finite, together with a total order $\leq$ on Q and three subsets

$$F_\forall, F_1, F_{>0} \subseteq Q \ .$$

A *run* of the automaton on an input tree $t : \{0,1\}^* \to \Sigma$ is an infinite binary tree $r : \{0,1\}^* \to Q$ whose root is labelled by the maximal state of Q, also called the *initial* state and which is consistent with the transition relation in the usual sense, i.e. $\forall v \in \{0,1\}^*, (r(v), t(v), r(v0), r(v1)) \in \Delta$. Define the *limsup* of a branch of the run to be the maximal state that appears infinitely often on the branch.

The run is *accepting* if it is surely, almost-surely and nonzero accepting:

- **surely accepting:** every branch has limsup in $F_\forall$; and
- **almost-surely accepting:** the set of branches with limsup in F_1 has probability 1; and
- **nonzero accepting:** for every node v with state in $F_{>0}$, the set of branches which visit v and visit only $F_{>0}$-labelled nodes below v has nonzero probability.

The emptiness problem. The emptiness problem asks whether an automaton has an accepting run. Our main result:

▶ **Theorem 3.** *Emptiness of a nonzero automaton is decidable in co-*NP.

Proof. This is a corollary of a series of intermediary results. In section 4 we focus on the special case where $F_\forall = Q$ (Theorem 10). In section 5 we reduce the emptiness problem for nonzero automata to the computation of the winner in a parity game called the *jumping game* (Lemma 17) and give an NP algorithm to compute the winner of the jumping game (Lemma 18). ◀

Zero automata. Nonzero automata are a variant of *zero automata* introduced in [1]. A zero automaton differs slightly from a nonzero automaton in that it uses a notion of "seed state" for the nonzero acceptance condition. On top of $F_\forall, F_1$ and $F_{>0}$ there is a subset $Q_{\text{seed}} \subseteq Q$. A run is accepting if it is surely, almost-surely and *zero* accepting:

- **zero accepting:** for every node v with state $q \in Q_{\text{seed}}$, with nonzero probability the run visits node v, then below v visits only states $\leq q$ and moreover has limsup in $F_{>0}$.

In the next section, we show that every zero automaton can be transformed in an equivalent nonzero automaton of quadratic size (Lemma 5). Combined with Theorem 3,

▶ **Corollary 4.** *The emptiness problem of zero automata is decidable in co-*NP*.*

According to [1], this implies that TMSO + zero has decidable satisfiability when zero is the unary predicate checking that a set of branches has probability 0.

An example: the dense but not very dense language. A tree over alphabet $\{a, b\}$ is *dense but not very dense* if:

1. every node has a descendant with label a; and
2. there is zero probability that a branch visit infinitely many nodes with letter a.

This language is non-empty, contains no regular tree and is recognized by a nonzero automaton. This automaton has three states, totally ordered as follows:

$$\underbrace{s}_{\text{searching for } a} < \underbrace{n}_{\text{not searching for } a} < \underbrace{f}_{\text{just found } a} .$$

The automaton begins in state f in the root. When the automaton reads a node with label b, then it sends s to some child and n to the other child, regardless of its current state. Choosing which child gets s and which child gets n is the only source of nondeterminism in this automaton. When the automaton sees letter a, it sends f to both children regardless of its current state. The acceptance condition is:

$$F_\forall = \{n, f\} \qquad F_1 = \{n\} \qquad F_{>0} = \emptyset \ .$$

3 From zero to nonzero automata

In this section we show that nonzero automata are as expressive as zero automata.

▶ **Lemma 5.** *For every zero automaton one can compute a nonzero automaton of quadratic size which accepts the same trees.*

The rest of the section is dedicated to the proof of Lemma 5, which is a direct corollary of Lemma 7 and Lemma 8 below.

Without loss of generality, we assume that in every *zero* automaton $F_{>0} \subseteq F_1 \subseteq F_\forall$. Changing F_1 for $F_1 \cap F_\forall$ and $F_{>0}$ for $F_{>0} \cap F_1$ does not modify the set of accepting runs of a zero automaton, since all branches should have limsup in $F_\forall$ and if the limsup is equal with nonzero probability to some $q \in F_{>0}$ then necessarily $q \in F_1$. By contrast, for *nonzero* automata there is no obvious reason for the same remark to hold.

We make use of an intermediary acceptance condition. Let r be a run. We say that a path from a node v to a node w is *seed-consistent* if whenever the path visits a seed state s, subsequent states are $\leq s$.

- **Strong zero acceptance condition:** for every node v labelled by a seed state, there is a seed-consistent path from v to a strict descendant w of v such that the state $r(w)$ of w is in $F_{>0}$ and there is a nonzero probability that the run
  - visits node w and visits only states $\leq r(w)$ below w,
  - has limsup $r(w)$,
  - in case $r(w) \notin Q_{\text{seed}}$, visits no seed state below w,
  - in case $r(w) \in Q_{\text{seed}}$, visits no seed state other than $r(w)$ below w.

Actually, the strong zero and zero acceptance conditions coincide (proof in appendix):

▶ **Lemma 6.** *A run is zero accepting if and only if it is strongly zero accepting.*

Construction of the nonzero automaton. Intuitively, every *zero* automaton can be simulated by a *nonzero* automaton which guesses on the fly a run of the zero automaton and checks simultaneously that the guessed run is strongly zero accepting. Whenever the automaton visits a node v with a seed state then it enters in the next step a *path-finding state* and guesses a seed-consistent path to a node w which is a witness of the strong zero condition. Once on the node w the automaton enters a *subtree-guessing* state and starts guessing a leaf-free subtree of the run, whose nodes are labelled by states $\leq r(w)$, whose branches have limsup $r(w)$ and which has nonzero probability.

There are some verifications to do in order to certify that the guessed run is strongly zero accepting. The surely accepting condition is used to prevent the automaton from staying forever in the path-finding mode and also to check that every branch of the subtree has limsup $r(w)$. The nonzero condition is used to check that the subtree has nonzero probability. To perform these verifications, the nonzero automaton stores some data in its control state. In path-finding mode the automaton records the smallest seed state seen so far in order to check on-the-fly that the path from v to w is seed-consistent. In subtree-guessing mode the automaton keeps track of the state $r(w)$.

The set of states of this automaton is denoted R, every state in R has as a first component a control state Q of the zero automaton. Precisely, R is the union of three sets:

- **normal states:** Q,
- **path-finding states:** $\{(q, s) \mid q \in Q, s \in Q_{\text{seed}}, q \leq s\}$,
- **subtree-guessing states:** $\{(q, f, *) \mid q \in Q, f \in F_{>0}, q \leq f, (q \notin Q_{\text{seed}} \vee q = f)\}$.

We equip R with any order $\prec$ such that

- the projection on the first component $\Pi_1 : (R, \prec) \to (Q, <)$ is monotonic,
- $(q, s) \prec q$ for every $q \in Q$ and $s \in Q_{\text{seed}}$ with $q \leq s$.

The zero, almost-surely and surely accepting conditions are defined respectively as:

$$G_{>0} = \text{ the set of subtree-guessing states,}$$
$$G_1 = F_1 \cup \{(f, f, *) \mid f \in F_{>0}\},$$
$$G_\forall = F_\forall \cup \{(f, f, *) \mid f \in F_{>0}\} \ .$$

The transitions of the automaton can be informally described as follows. The nonzero automaton guesses on the fly a run $\rho : \{0, 1\}^* \to Q$ of the zero automaton by storing the value of $\rho(v)$ as the first component of its own control state on the node v. The nonzero automaton stays in the set of normal states as long as the run does not enter a seed state. On a node v labelled by $s \in Q_{\text{seed}}$, the nonzero automaton starts looking for a path to a descendant node w that satisfies the strong zero condition. For that in the next step the automaton enters either a path-finding or a subtree-guessing state. While in a path-finding state, the

automaton guesses on the fly a seed-consistent path. Whenever the run is in a nonzero state $f \in F_{>0}$ the nonzero automaton can enter the subtree-guessing state $(f, f, *)$, or not. While in subtree-guessing mode the second component is constant, and the automaton control state is of type $(q, f, *)$ with $q \leq f$ and $q \notin Q_{\text{seed}}$ unless $q = f \in Q_{\text{seed}}$. From a subtree-guessing state the automaton may switch back any time to a normal state.

Formally, for every transition $q \to r_0, r_1$ of the zero automaton, there is a transition

$$q' \to r'_0, r'_1$$

in the nonzero automaton if the first component of q' is q and

$$r'_0 = \begin{cases} r_0 & \text{whenever } q' \text{ is not path-finding} \\ (r_0, r_0, *) & \text{whenever } \begin{cases} q \in Q_{\text{seed}}, q' = q \text{ and } r_0 \in F_{>0} \text{ and } r_0 \leq q \\ \text{or } q' = (q, s) \text{ and } r_0 \in F_{>0} \text{ and } r_0 \leq s, \end{cases} \\ (r_0, f, *) & \text{whenever } q' = (q, f, *) \text{ and } r_0 \leq f \text{ and } (r_0 \notin Q_{\text{seed}} \vee r_0 = f). \end{cases}$$

The possible values of r'_1 are symmetric. There are also *left path-finding transitions*: for every seed states $s, s' \in Q_{\text{seed}}$ such that $q \leq s$ and $r_0 \leq s$ there are transitions

$$q' \to (r_0, s'), r_1 \text{ where } q' = \begin{cases} q \text{ or } (q, q) \text{ if } q = s \\ (q, s) \text{ otherwise} \end{cases} \quad \text{and } s' = \begin{cases} s & \text{if } r_0 \notin Q_{\text{seed}} \\ r_0 & \text{if } r_0 \in Q_{\text{seed}}. \end{cases}$$

There may also be a symmetric *right path-finding transition* $(q, s) \to r_0, (r_1, s')$ when the symmetric conditions hold.

The next two lemmas relate the accepting runs of the zero and the nonzero automata, their proofs can be found in the appendix.

▶ **Lemma 7.** *Let $d : \{0,1\}^* \to R$ be an accepting run of the nonzero automaton. Then its projection $r = \Pi_1(d)$ on the first component is an accepting run of the zero automaton.*

▶ **Lemma 8.** *If the zero automaton has an accepting run $r : \{0,1\}^* \to Q$ then the nonzero automaton has an accepting run $d : \{0,1\}^* \to R$ such that $r = \Pi_1(d)$.*

4 Emptiness of $F_\forall$-trivial automata is in NP

A run of a nonzero automaton needs to satisfy simultaneously three conditions, which correspond to the accepting sets $F_\forall, F_1, F_{>0}$. For a subset

$$I \subseteq \{F_\forall, F_1, F_{>0}\}$$

define I-automata to be the special case of nonzero automata where only the acceptance conditions corresponding to I need to be satisfied. These are indeed special cases: ignoring $F_{>0}$ can be achieved by making it empty, ignoring F_1 can be achieved by making it equal to $F_\forall$, and ignoring $F_\forall$ can be achieved by making it equal to all states Q.

Generalizing parity automata, with standard and qualitative semantics. A $\{F_\forall\}$-automaton is a parity automaton. Thus solving emptiness for nonzero automata is at least as hard as emptiness for parity automata on trees, which is polynomial time equivalent to solving parity games, in NP ∩co-NP [12] or in quasi-polynomial time [4].

A $\{F_1\}$-automaton is the same as a parity automaton with qualitative semantics as introduced in [5]. Emptiness for such automata can be solved in polynomial time using standard linear programming algorithms for Markov decision processes.

Subzero automata. A $\{F_1, F_\forall\}$-automaton is the same as a *subzero* automaton as considered in [10]. In [10], it was shown how to decide if a subzero automaton accepts some regular tree. Since some subzero automata are nonempty but accept no regular trees, see e.g. the example in [1], the result from [10] does not solve non-emptiness for subzero automata.

$F_\forall$-trivial automata. In a $\{F_1, F_{>0}\}$-automaton, the surely accepting condition is trivial, i.e. $F_\forall = Q$. We call such automata $F_\forall$-trivial. The acceptance of a run of a $F_\forall$-trivial automaton depends only on the probability measure on Q^ω induced by the run, individual branches do not matter.

▶ **Definition 9** (Positional run)**.** A run is *positional* if whenever the states of two nodes coincide then the states of their left children coincide and the states of their right children coincide.

▶ **Theorem 10.** *If a $F_\forall$-trivial automaton has an accepting run, then it has a positional accepting run. Emptiness of $F_\forall$-trivial automata is in co-*NP*.*

This result was recently improved in [2]: the complexity is actually PTIME. The proof of this theorem relies on the notion of acceptance witnesses.

▶ **Definition 11** (Transition graph and acceptance witness)**.** Let D be a set of transitions.

The transition graph of D, denoted G_D, is the directed graph whose vertices are all states appearing in one of the transitions in D, denoted Q_D, and whose edges are induced by the transitions in D: for every $(q, a, l, r) \in D$ both (q, l) and (q, r) are edges of G_D.

The set D is an *acceptance witness* if it satisfies the four following conditions:

(i) Q_D contains the initial state of the automaton and G_D has no dead-end,
(ii) the maximum of every bottom strongly connected component (BSCC) of G_D is in F_1,
(iii) every BSCC of G_D is either contained in $F_{>0}$ or does not intersect $F_{>0}$,
(iv) from every state in $F_{>0} \cap Q_D$ there is a path in $F_{>0} \cap Q_D$ to a BSCC contained in $F_{>0}$.

▶ **Lemma 12.** *If a $F_\forall$-trivial automaton has an acceptance witness, it has a positional accepting run.*

Proof. The proof is by induction on $N_D = |D| - |Q_D|$. Since G_D has no dead-end, every state in Q_D is the source of a transition in D thus $N_D \geq 0$.

If $N_D = 0$ then for every state $q \in Q_D$ there is a unique transition $\delta_q = (q, a_q, l_q, r_q)$. Let ρ be the positional run whose root has the initial state and every node with vertex $q \in Q_D$ has children l_q and r_q, which is well-defined according to property (i). We show that ρ is an accepting run. The graph G_D can be seen as a Markov chain, with probability either 1 or $\frac{1}{2}$ on every edge, depending on the out-degree. The probability measure on Q_D^ω produced by the random walk on ρ coincide with the probability measure on Q_D^ω produced by this finite Markov chain: indeed both measures coincide on finite cylinders $q_0 \cdots q_n Q_D^\omega$. Basic theory of finite homogenous Markov chain implies that almost-surely every branch of the run ends up in one of the BSCCs of G_D and visits all its states infinitely often. Thus property (ii) ensures that the run ρ is almost-surely accepting. Properties (iii) and (iv) guarantee that the run is moreover nonzero-accepting.

Assume now that $N_D > 0$. We show that there is a strictly smaller acceptance witness $D' \subsetneq D$. Let $q \in Q_D$ which is the source of several transitions in D, then D' is obtained by removing from D all these transitions except one. To choose which transition δ to keep, pick some shortest path $q = q_0 \dots q_n$ in G_D of length ≥ 1 which leads to a maximal state of one of the BSCCs of G_D. Moreover if $q \in F_{>0}$ we require the whole path to stay in $F_{>0}$. By

definition of G_D there is at least one transition in D whose origin is q and one of the two successors is q_1. To get D' we delete all other transitions with source q from D.

Clearly property (i) is preserved by this operation. To address properties (ii)–(iv), we show that every BSCC B' of $G_{D'}$ is either a BSCC of G_D or contained in the BSCC B of G_D whose maximum is q_n, in which case $\max B = \max B' = q_n$. There are two cases. If B' does not contain q_n then it does not contain q either (because $q = q_0 \dots q_n$ is still a path in $G_{D'}$). Since the only difference between G_D and $G_{D'}$ are the outgoing transitions from q then B' is actually a BSCC of G_D. If B' contains q_n then $B' \subseteq B$ (because there are less edges in $G_{D'}$ than in G_D) and since $q_n = \max B$ then $\max B = \max B'$.

As a consequence property (ii) and (iii) are preserved. And property (iv) is preserved as well: in case $q \notin F_{>0}$ then there is nothing to prove and in case $q \in F_{>0}$ then $q = q_0 \dots q_n$ is still a path in $G_{D'}$, with all vertices in $F_{>0}$. Moreover the set of vertices from which q_n is accessible is the same in G_D and $G_{D'}$ thus q_n is in a BSCC of $G_{D'}$. ◀

A strong version of the converse implication of Lemma 12 holds:

▶ **Lemma 13.** *If a $F_\forall$-trivial automaton has an accepting run, it has an acceptance witness.*

Proof. We fix an accepting run ρ on some input tree t. To extract an acceptance witness from ρ, we make use of the notion of end-component introduced in [6].

▶ **Definition 14** (End-component)**.** The *transition of a node v* is $d(v) = (\rho(v), t(v), \rho(v0), \rho(v1))$. For every branch b, we denote $\Delta^\infty(b)$ the set of transitions labelling infinitely many nodes of the branch. For every subset $D \subseteq \Delta$ we denote B_D the set of branches b such that $\Delta^\infty(b) = D$. A set of transitions $D \subseteq \Delta$ is an *end-component* of the run if B_D has nonzero probability.

Call a branch b *even* if for every transition $\delta = (q, a, l, r) \in \Delta^\infty(b)$, not only the state q but also the states l and ρ appear infinitely often on the branch in the run ρ. Almost-surely every branch is even, because each time a branch visits a node with transition δ it proceeds left or right with equal probability $\frac{1}{2}$. As a consequence,

▶ **Lemma 15.** *Let D be an end-component of the run. Then the transition graph of D has no dead-end, is strongly connected and its maximal state is in F_1.*

Proof. Denote G_D the transition graph of D, with states Q_D. Since D is an end-component then B_D has non-zero probability, and since almost every branch is even then B_D contains at least one even branch b. The set of states appearing infinitely often on b is exactly Q_D. By removing a prefix long enough of b so that only states in Q_D occur on the remaining suffix then one obtains a path in G_D which visits every state in Q_D infinitely often. Thus G_D has no dead-end and is strongly connected. Moreover every even branch in B_D has limsup $\max Q_D$ and since the run is almost-surely accepting then $\max Q_D \in F_1$. ◀

Let $\mathcal{D}$ be the collection of all end-components of the run ρ. We define two subsets of $\mathcal{D}$, denoted respectively $\mathcal{D}_0$ and $\mathcal{D}_1$, which collect the end-components whose states are respectively included in $F_{>0}$ and disjoint from $F_{>0}$. Let $D_0 \subseteq \Delta$ (resp. $D_1 \subseteq \Delta$) be the union of all end-components in $\mathcal{D}_0$ (resp. in $\mathcal{D}_1$). These transitions are easy to reach:

▶ **Lemma 16.** *Every node v has a descendant w whose transition $d(w)$ belongs to $D_0 \cup D_1$. Moreover if the state of v is in $F_{>0}$ then w can be chosen such that the path v to w is labelled by $F_{>0}$ and the transition $d(w)$ is in D_0.*

Proof. Let v be a node and S_v the set of branches which visit v and, in case v is labelled by $F_{>0}$, visit only $F_{>0}$-labelled nodes below v. Since the run is accepting then S_v has positive probability. By definition of end-components, almost-every branch is in $\bigcup_{D\in\mathcal{D}} B_D$. Thus there exists an end-component D such that $B_D \cap S_v$ has positive probability. As a consequence, v has a descendant w whose transition is in D. Since almost-every branch is even and $B_D \cap S_v$ has positive probability then there is at least one branch in $B_D \cap S_v$ which visits infinitely often all states appearing in Q_D. In case v is labelled by $F_{>0}$, this implies that $Q_D \subseteq F_{>0}$ thus $D \in \mathcal{D}_0$, and the proof of the second statement is complete. In case v has no descendant labelled by $F_{>0}$ this implies that $Q_D \cap F_{>0} = \emptyset$ thus $D \in \mathcal{D}_1$, and the first statement holds in this case. In the remaining case, v has a descendant v' labelled with $F_{>0}$, which itself has a descendant w whose transition belongs to some $D \in \mathcal{D}_0$, thus the first statement holds for v. ◀

We terminate the proof of Lemma 13. Let G_0 (resp. G_1) the transition graph of D_0 (resp. D_1) and denote Q_0 (resp. Q_1) the set of states of G_0 (resp. G_1).

Let D be the set of all transitions appearing in the run. According to Lemma 16, in the transition graph G_D, $Q_0 \cup Q_1$ is accessible from every state $q \in Q_D$ and moreover Q_0 is accessible from every state $q \in Q_D \cap F_{>0}$ following a path in $Q_D \cap F_{>0}$.

We say that an edge (q, r) of G_D is *progressive* if $q \notin Q_0 \cup Q_1$ and either ($q \in F_{>0}$ and $r \in F_{>0}$ and (q, r) decrements the distance to Q_0 in G_D) or ($q \notin F_{>0}$ and (q, r) decrements the distance to $Q_0 \cup Q_1$ in G_D). Every state in $Q_D \setminus (Q_0 \cup Q_1)$ is the source of at least one progressive edge.

We denote D_+ the union of D_0 and D_1 plus all the transitions $\delta = (q, a, r_0, r_1) \in D$ such that either (q, r_0) or (q, r_1) is progressive. Then D_+ has all four properties of Lemma 12. Denote G_+ the transition graph associated to D_+. Property (i) holds because every state in Q_D, including the initial state, is either in $Q_0 \cup Q_1$ or is the source of a progressive edge.

Remark that the BSCCs of G_+ are exactly the BSCCs of G_0 and G_1. Since both G_0 and G_1 are unions of strongly connected graphs, they are equal to the union of their BSCCs. The BSCCs of G_0 and G_1 are still BSCCs in G_+ because no edges are added inside them (progressive edges have their source outside G_0 and G_1). Following the progressive edges leads to G_0 or G_1 from every state in G_+, thus there are no other BSCCs in G_+.

This implies property (ii) because, according to Lemma 15, both graphs G_0 and G_1 are the union of strongly connected graphs whose maximal states are in F_1. This also implies property (iii) since $Q_0 \subseteq F_{>0}$ and $Q_1 \cap F_{>0} = \emptyset$. Property (iv) is obvious for states in Q_0 because Q_0 is a union of BSCCs included in $F_{>0}$. Property (iv) holds as well for states in $(Q_D \cap F_{>0}) \setminus Q_0$, the path to Q_0 is obtained following the progressive edges in $F_{>0} \times F_{>0}$. ◀

Proof of Theorem 10. According to Lemma 13, the existence of an accepting run implies the existence of an acceptance witness and according to Lemma 12 this implies the existence of a positional accepting run. Guessing a subset of transitions and checking it is an acceptance witness can be done in non-deterministic polynomial time. ◀

5 Emptiness of nonzero automata is in co-NP

In this section we show how to decide the emptiness of nonzero automata. The main ingredient are jumping games.

Call a run $\{F_1, F_{>0}\}$-*accepting* if it satisfies the almost-surely and the nonzero acceptance condition, but it does not necessarily satisfy the surely accepting condition, and the condition on the initial state is dropped as well.

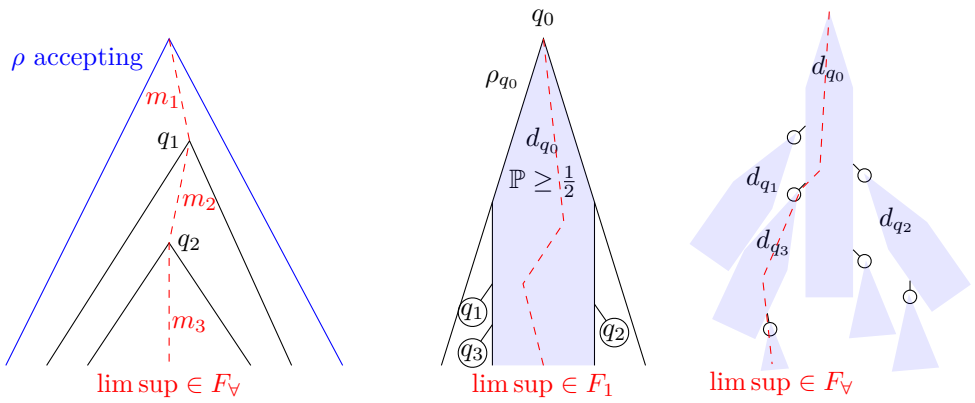


Figure 1 The left picture illustrates how an accepting run is turned into a winning strategy for Automaton in the jumping game, the two other pictures illustrate the converse transformation.

The jumping game. For a run ρ, define its *profile* to be the set of state pairs (q, m) such that some non-root node in ρ has state q and m is the maximal state of its strict ancestors.

The *jumping game* is a parity game played by two players, *Automaton* and *Pathfinder*. Positions of Automaton are states of the automaton and positions of Pathfinder are profiles of $\{F_1, F_{>0}\}$-accepting runs, not necessarily positional. The game is an edge-labelled parity game, i.e. the priorities are written on the edges. The edges originating in Automaton positions are of the form

$$q \xrightarrow{q} \Pi \qquad \text{such that } \Pi \text{ is the profile of some } \{F_1, F_{>0}\}\text{-accepting run with root state } q.$$

The edges originating in Pathfinder positions are of the form

$$\Pi \xrightarrow{m} q \qquad \text{such that } (q, m) \in \Pi.$$

We say that Automaton wins the jumping game if he has a winning strategy from the position which is the initial state of the automaton. If the play ever reaches a dead-end, i.e. a state which is not the root of any $\{F_1, F_{>0}\}$-accepting run, then the game is over and Automaton loses. Otherwise Automaton wins iff the limsup of the priorities is in $F_\forall$.

Lemmas 17 and 18 below establish that non-emptiness of a nonzero automaton is equivalent to Automaton winning the jumping game, and this can be decided in NP.

▶ **Lemma 17.** *The automaton is nonempty if and only if Automaton wins the jumping game.*

Sketch of Proof. The proof transforms an accepting run ρ of the nonzero automaton into a winning strategy σ of Automaton, and back, this is illustrated by Fig. 1.

When the nonzero automaton has an accepting run ρ, Automaton can win the jumping game as follows. For a start, Automaton plays the profile Π_0 of ρ. Then Pathfinder chooses some pair $(q_1, m_1) \in \Pi_0$, by definition of profiles this corresponds to some non-root node v_1 of ρ labelled by q_1 and m_1 is the maximal state of the ancestors of v_1. At each step $n > 0$, Pathfinder chooses some pair $(q_n, m_n) \in \Pi_n$ corresponding to some node v_{n+1} whose v_n is a strict ancestor, and Automaton plays the profile Π_n of the subtree ρ_n of ρ rooted in v_n. Since the run ρ is accepting then *a fortiori* the run ρ_n is $\{F_1, F_{>0}\}$-accepting. Quite clearly, this is a winning strategy for Automaton.

Conversely, we use a positional winning strategy of Automaton (whose existence is well-known [12]) to build an accepting run of the nonzero automaton. Denote W the set of states

winning for Automaton. With every state q in W we associate the profile Π_q chosen by the positional winning strategy of Automaton and a $\{F_1, F_{>0}\}$-accepting run ρ_q with profile Π_q.

We show the existence of a leaf-free subtree d_q of ρ_q such that:

(a) the set of branches of d_q has probability $\geq \frac{1}{2}$,

(b) *every* branch of d_q has limsup in F_1,

(c) for every node v of d_q with state in $F_{>0}$, the set of branches of d_q which visit v and visit only $F_{>0}$-labelled nodes below v has nonzero probability.

Since ρ_q is almost-surely accepting, then according to Lemma 1, there is a subtree d_q of ρ_q whose set of branches has probability $\geq \frac{1}{2}$ and *all of them* have limsup in F_1 (while in the run ρ_q there may be a non-empty set of branches with limsup in $F_\forall \setminus F_1$, with probability zero). Since we are only interested in branches of d_q, we can assume that d_q is leaf-free. This guarantees properties a) and b) but not c), however using Lemma 1 again, we can extend d_q to d'_q such that property c) holds as well.

These partial runs $(d'_q)_{q\in W}$ can be combined in order to get a graph whose unravelling, starting from the initial state, is an accepting run of the automaton. Each time a branch enters a subtree d'_q, there is probability $\geq \frac{1}{2}$ to stay in d_q forever. Thus almost every branch of the unravelling eventually stays in one of the subtrees $(d'_q)_{q\in W}$, thus has limsup in $F_1 \subseteq F_\forall$ according to property b). As a consequence the unravelling is almost-surely accepting. Still, with probability 0, some branches switch infinitely often from a subtree to another. These branches correspond to an infinite play consistent with σ and are $F_\forall$-accepting. ◀

▶ **Lemma 18.** *Given a nonzero automaton, whether Automaton wins the jumping game is decidable in* NP .

The game is not constructed explicitly, which would require exponential time, but strategies of Automaton can be represented in a compact way, which is enough to get the NP upper bound. This result was recently improved: the winner can be decided in NP∩co-NP, see [2].

Sketch of Proof. By positional determinacy of parity games, it suffices to find a positional strategy of player Automaton, which maps states to profiles of $\{F_1, F_{>0}\}$-accepting runs. It is equivalent and easier to find an *acceptance witness*. This is a pair (W, σ) where W is a subset of Q containing the initial state of the automaton, and $\sigma : W \to 2^{W\times W}$ satisfies:

α) For every sequence $(q_0, m_0)(q_1, m_1)\dots$ in $(W \times W)^\omega$, if q_0 is the initial state of the automaton and $\forall n, (q_{n+1}, m_{n+1}) \in \sigma(q_n)$ then $\limsup_n m_n \in F_\forall$.

β) $\forall q \in W$, $\sigma(q)$ *contains* the profile of a $\{F_1, F_{>0}\}$-accepting run with initial state q.

Finding a witness can be done in NP. Condition α) is checked in linear time. Given $\Pi \subseteq Q\times Q$, one can use Theorem 10 to check condition Condition β) in NP, by storing in the state space of the automaton the maximal state of the ancestors of the curent node. ◀

Example: the everywhere positive language. A tree t on the alphabet $\{a, b\}$ is *everywhere positive* if for every node v,

1. there is positive probability to see only the letter $t(v)$ below v,
2. there is positive probability to see finitely many times the letter $t(v)$ below v.

This language is non-empty and contains no regular tree. The language of everywhere positive trees with root state a is recognized by a nonzero automaton with six states

$$\{s_b < s_a < n_b < n_a < f_b < f_a\} \quad .$$

On a node labelled by letter a, the automaton can perform a transition from any of the three states $\{s_b, n_b, f_a\}$, meaning intuitively "searching for b", "not searching for b" and "just

found a". From these states the automaton can choose any pair of successor states which intersects $\{s_b, f_b\}$. Transitions on letter b are symmetrical. The acceptance condition is:

$$F_\forall = \{n_a, n_b, f_a, f_b\} \qquad F_1 = F_\forall \qquad F_{>0} = \{n_a, s_a, n_b, s_b\} \ .$$

Among the simplest moves of Automaton in the jumping game are the two moves $n_b \to \{(n_b, n_b)(s_b, n_b)\}$ and $s_b \to \{(n_b, n_b)(s_b, n_b)\}$, which correspond to the profiles of some $\{F_1, F_{>0}\}$-accepting runs on the tree whose all nodes have letter a, and everywhere in the tree the automaton applies the same two transitions $n_b \to_b (n_b, s_b)$ and $s_b \to_b (n_b, s_b)$. In those runs, the automaton always looks for a letter b in the right direction (state s_b), and does not look for b in the left direction (state n_b). Since the tree has no b at all then the quest for a letter b is hopeless, and on all branches of the run that ultimately always turn right (i.e. branches in $\{0,1\}^*1^\omega$), the automaton ultimately stays in state s_b and the branch has limsup s_b, which is neither in $F_\forall$ nor in F_1. But such branches happen with probability zero: almost-every branch performs infinitely many turns left and right, thus has limsup n_b. As a consequence such a run is almost-surely accepting: Such a run is nonzero-accepting as well because every node labelled by $F_{>0}$ has all its descendants labelled by $F_{>0}$.

Yet legal, these two moves are not good options for Automaton in the jumping game because then Pathfinder can generate the play

$$s_b \xrightarrow{s_b} \{(n_b, n_b)(s_b, n_b)\} \xrightarrow{n_b} s_b \xrightarrow{s_b} \{(n_b, n_b)(s_b, n_b)\} \xrightarrow{s_b} s_b \xrightarrow{s_b} \dots$$

which has limsup $n_b = \max\{s_b, n_b\}$ and is losing for Automaton since $n_b \notin F_\forall$.

Actually, Automaton can win with the moves

$$s_a/n_a \to \{(f_a, f_a), (n_b, f_a), (s_b, f_a), (n_a, n_a), (s_a, n_a)\}$$
$$f_a \to \{(n_b, f_a), (s_b, f_a)\}$$

and their symmetric counterparts from states $\{s_b, n_b, f_b\}$. In the jumping game, this forces Pathfinder to take only edges labelled by one of the states $\{f_a, n_a, f_b, n_b\}$. These states dominate the states $\{s_a, s_b\}$ thus the limsup of the corresponding plays is in $F_\forall$ and Automaton wins.

6 Conclusion

We have shown that the emptiness problem for zero and nonzero automata is decidable and belongs to co-NP. As a consequence, the satisfiability for the logic MSO+zero from [1] is decidable (in non-elementary time), when zero is the unary predicate that checks a set of branches has probability 0.

As shown by Stockmeyer, the satisfiability problem for first-order logic on finite words cannot be solved in elementary time [11]. Therefore any translation from a logic stronger than first-order logic on finite words (such as TMSO+zero on infinite trees) to an automaton model with elementary emptiness (such as nonzero automata) is necessarily non-elementary. This does not make the relatively low NP complexity of nonzero automata any less interesting. One can imagine other logics than TMSO+zero, either less expressive or maybe even equally expressive but less succinct, which will have a relatively low complexity by virtue of a translation into nonzero automata. One natural direction is the study of temporal logics.

Our results were recently improved [2]: the emptiness of nonzero automata actually belongs to NP∩co-NP, and is even in PTIME for $F_\forall$-trivial automata.

Acknowledgments. We thank Henryk Michalewski and Matteo Mio for helpful discussions.

References

1. Mikolaj Bojanczyk. Thin MSO with a Probabilistic Path Quantifier. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 96:1–96:13. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.96.
2. Mikołaj Bojańczyk, Hugo Gimbert, and Edon Kelmendi. Emptiness of zero automata is decidable. Technical report, CNRS, 2017. URL: http://arxiv.org/abs/1702.06858.
3. Tomás Brázdil, Vojtech Forejt, and Antonín Kucera. Controller synthesis and verification for mdps with qualitative branching time objectives. In *ICALP 2008.*, pages 148–159, 2008.
4. C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. Technical report, CDMTCS, October 2016. URL: https://www.cs.auckland.ac.nz/research/groups/CDMTCS/researchreports/index.php?download&paper_file=631.
5. Arnaud Carayol, Axel Haddad, and Olivier Serre. Randomization in automata on infinite trees. *ACM Trans. Comput. Log.*, 15(3):24:1–24:33, 2014. doi:10.1145/2629336.
6. L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, december 1997.
7. A. S. Kechris. *Classical Descriptive Set Theory.* Graduate Texts in Mathematics. Springer-Verlag, 1995.
8. Daniel Lehmann and Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–1983, 1982.
9. Henryk Michalewski and Matteo Mio. Measure quantifier in monadic second order logic. In *LFCS 2016, Deerfield Beach, FL, USA, January 4-7, 2016. Proceedings*, pages 267–282, 2016. doi:10.1007/978-3-319-27683-0_19.
10. Henryk Michalewski, Matteo Mio, and Mikołaj Bojańczyk. On the regular emptiness problem of subzero automata. *CoRR*, abs/1608.03319, 2016. URL: http://arxiv.org/abs/1608.03319.
11. Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic.* PhD thesis, MIT, 1974.
12. Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.

Characterizing Definability in Decidable Fixpoint Logics*

Michael Benedikt[1], Pierre Bourhis[2], and Michael Vanden Boom[3]

1 **Department of Computer Science, University of Oxford, Oxford, UK**
2 **CNRS CRIStAL UMR 9189, INRIA Lille, Lille, FR**
3 **Department of Computer Science, University of Oxford, Oxford, UK**

Abstract

We look at characterizing which formulas are expressible in rich decidable logics such as guarded fixpoint logic, unary negation fixpoint logic, and guarded negation fixpoint logic. We consider semantic characterizations of definability, as well as effective characterizations. Our algorithms revolve around a finer analysis of the tree-model property and a refinement of the method of moving back-and-forth between relational logics and logics over trees.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Guarded logics, bisimulation, definability, automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.107

1 Introduction

A major line of research in computational logic has focused on obtaining extremely expressive decidable logics. The guarded fragment (GF) [1], the unary negation fragment (UNF) [23], and the guarded negation fragment (GNF) [3] are rich decidable fragments of first-order logic. Each of these has extensions with a fixpoint operator that retain decidability: GFP [18], UNFP [23], and GNFP [3] respectively. In each case the argument relies on "moving to trees". This involves showing that the logic possesses the tree-like model property: whenever there is a satisfying model for a formula, it can be taken to be of tree-width that can be effectively computed from the formula. Such models can be coded by trees, thus reducing satisfiability of the logic to satisfiability of a corresponding formula over trees, which can be decided using automata-theoretic techniques. This method has been applied for decades (e.g. [25, 16]).

A question is how to recognize formulas in these logics, and more generally how to distinguish the properties of the formulas in one logic from another. Clearly if we start with a formula in an undecidable logic, such as first-order logic or least fixed point logic (LFP), we have no possibility for effectively recognizing any non-trivial property. But we could still hope for an insightful semantic characterization of the subset that falls within the decidable logic. One well-known example of this is van Benthem's theorem [24] characterizing modal logic within first-order logic – a first-order sentence is equivalent to a modal logic sentence exactly when it is bisimulation-invariant. For fixpoint logics, an analogous characterization is the Janin-Walukiewicz theorem [20], stating that the modal mu-calculus (L_μ) captures the bisimulation-invariant fragment of monadic second-order logic (MSO). If we start in one decidable logic and look to characterize another decidable logic, we could hope for a

* Benedikt and Vanden Boom were funded by the EPSRC grants PDQ (EP/M005852/1), ED^3 (EP/N014359/1), and DBOnto (EP/L012138/1). Bourhis was funded by the DeLTA project (ANR-16-CE40-0007).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 107; pp. 107:1–107:14

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

characterization that is effective. For example, Otto [22] showed that if we start with a formula of L_μ, we can determine whether it can be expressed in modal logic.

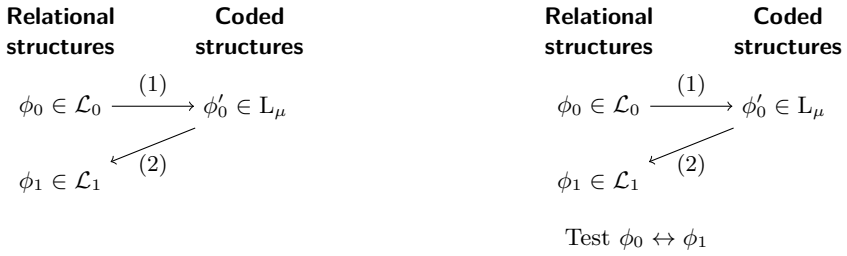
In this work we will investigate both semantic and effective characterizations. We will begin with GFP. Grädel, Hirsch, and Otto [16] have already provided a characterization of GFP-definability within a very rich logic extending MSO called guarded second-order logic (GSO). The characterization is exactly analogous to the van Benthem and Janin-Walukiewicz results mentioned above: GFP captures the "guarded bisimulation-invariant" fragment of GSO. The characterization makes use of a refinement of the method used for decidability of these logics, which moves *back-and-forth between relational structures and trees*: (1) define a *forward mapping* taking a formula ϕ_0 in the larger logic (e.g. GSO invariant under guarded bisimulation) over relational structures to a formula ϕ'_0 over trees that describes codes of structures satisfying ϕ_0; (2) define a *backward mapping* based on the invariance going back to some ϕ_1 in the restricted logic (e.g. GFP). The method is shown in Figure 1a.

Our first main theorem is an effective version of the above result: if we start with a formula in certain richer decidable fixpoint logics, such as GNFP, we can decide whether the formula is in GFP. At the same time we provide a refinement of [16] which accounts for two signatures, the one allowed for arbitrary relations and the one allowed for "guard relations" that play a key role in the syntax of all guarded logics. We extend this result to deciding membership in the "k-width fragment", GNFP^k; roughly speaking this consists of formulas built up from guarded components and positive existential formulas with at most k variables. We provide a semantic characterization of this fragment within GSO, as the fragment closed under the corresponding notion of bisimulation (essentially, the GN^k-bisimulation of [3]). As with GFP, we show that the characterization can be made effective, provided that one starts with a formula in certain larger decidable logics. The proof also gives an effective characterization for the k-width fragment of UNFP.

As in the method for invariance and decidability above, we apply a forward mapping to move from a formula ϕ_0 in a larger logic $\mathcal{L}_0$ on relational structures to a formula ϕ'_0 on tree encodings. But then we can apply a *different backward mapping*, tuned towards the smaller logic $\mathcal{L}_1$ and the special properties of its tree-like models. The backward mapping of a tree property ϕ'_0 is always a formula ϕ_1 in the smaller logic $\mathcal{L}_1$ (e.g. GFP). But it is no longer guaranteed to be "correct" unconditionally – i.e. to always characterize structures whose codes satisfy ϕ'_0. Still, we show that *if* the original formula ϕ_0 is definable in the smaller logic $\mathcal{L}_1$, then the backward mapping applied to the forward mapping gives such a definition. Since we can check equivalence of two sentences in these logics effectively, this property suffices to get decidability of definability. The revised method is shown schematically in Figure 1b.

The technique above has a few inefficiencies; first, the general forward mapping passes through MSO and has non-elementary complexity. Secondly, the technique implicitly moves between relational structures and trees *twice*: once to construct ϕ_0, and a second time to check that ϕ_0 is equivalent to ϕ_1, which in turn requires first forming a formula ϕ'_1 over trees via a forward mapping and then checking its equivalence with ϕ'_0. We show that in some cases we can optimize this, allowing us to get tight bounds on the equivalence problem.

We show that our results "restrict" to fragments of these guarded logics, including their first-order fragments. In particular, our results give effective characterizations of GF definability when the input is in FO. They can be thus seen as a generalization of well-known effective characterizations of the conjunctive existential formulas in GF, the *acyclic queries*. We show that we can apply our techniques to the problem of transforming conjunctive formulas to a well-known efficiently-evaluable form (acyclic formulas) relative to GF theories. These results complement previous results on query evaluation with constraints from [6, 13].



(a) Back-and-forth for Semantic Characterization.

(b) Back-and-forth for Effective Characterization.

Figure 1

This refined back-and-forth method can be tuned in a number of ways, allowing us to control the signature as well as the sublogic. We show this can be adapted to give an approximation of the formula ϕ_0 within the logic $\mathcal{L}_1$, which is a kind of *uniform interpolant*.

2 Preliminaries

We work with finite relational signatures σ. We use $\boldsymbol{x}, \boldsymbol{y}, \dots$ (respectively, $\boldsymbol{X}, \boldsymbol{Y}, \dots$) to denote vectors of first-order (respectively, second-order) variables. For a formula ϕ, we write $\phi(\boldsymbol{x})$ to indicate that the free first-order variables in ϕ are among $\boldsymbol{x}$. If we want to emphasize that there are also free second-order variables $\boldsymbol{X}$, we write $\phi(\boldsymbol{x}, \boldsymbol{X})$. We often use α to denote atomic formulas, and if we write $\alpha(\boldsymbol{x})$ then we assume that the free variables in α are precisely $\boldsymbol{x}$. The *width* of ϕ, denoted $\mathrm{width}(\phi)$, is the maximum number of free variables in any subformula of ϕ, and the width of a signature σ is the maximum arity of its relations.

The *Guarded Negation Fragment* of FO [3] (denoted GNF) is built up inductively according to the grammar $\phi ::= R\,\boldsymbol{x} \mid \exists x.\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \alpha(\boldsymbol{x}) \wedge \neg\phi(\boldsymbol{x})$ where R is either a relation symbol or the equality relation, and α is an atomic formula or equality such that $\mathrm{free}(\alpha) \supseteq \mathrm{free}(\phi)$. Such an α is a *guard*. If we restrict α to be an equality, then each negated formula can be rewritten to use at most one free variable; this is the *Unary Negation Fragment*, UNF [23]. GNF is also related to the *Guarded Fragment* [1] (GF), typically defined via the grammar $\phi ::= R\,\boldsymbol{x} \mid \exists \boldsymbol{x}.\alpha(\boldsymbol{xy}) \wedge \phi(\boldsymbol{xy}) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(\boldsymbol{x})$ where R is either a relation symbol or the equality relation, and α is an atomic formula or equality that uses all of the free variables of ϕ. Here it is the quantification that is guarded, rather than negation. GNF subsumes GF sentences and UNF formulas.

The fixpoint extensions of these logics (denoted GNFP, UNFP, and GFP) extend the base logic with formulas $[\mathbf{lfp}_{X,\boldsymbol{x}}.\alpha(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{x})$ where (i) $\alpha(\boldsymbol{x})$ is an atomic formula or equality guarding $\boldsymbol{x}$, (ii) X only appears positively in ϕ, (iii) second-order variables like X cannot be used as guards. Some alternative (but equi-expressive) ways to define the fixpoint extension are discussed in [3]; in all of the definitions, the important feature is that tuples in the fixpoint are guarded by an atom in the original signature. In UNFP, there is an additional requirement that only unary or 0-ary predicates can be defined using the fixpoint operators. GNFP subsumes both GFP sentences and UNFP formulas. These logics are all contained in LFP, the fixpoint extension of FO, so the semantics are inherited from there.

It is often helpful to consider the formulas in a normal form. *Strict normal form* GNFP

formulas can be generated using the following grammar:

$$\phi ::= \bigvee_i \exists \boldsymbol{x}_i . \bigwedge_j \psi_{ij}$$
$$\psi ::= R\,\boldsymbol{x} \mid X\,\boldsymbol{x} \mid \alpha(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}) \mid \alpha(\boldsymbol{x}) \wedge \neg\phi(\boldsymbol{x}) \mid [\mathbf{lfp}_{X,\boldsymbol{x}} .\alpha(\boldsymbol{x}) \wedge \phi(\boldsymbol{x}, X, \boldsymbol{Y})](\boldsymbol{x})$$

where α is an atomic formula or equality statement such that $\text{free}(\alpha) = \text{free}(\phi)$; we call such an α a *strict guard*. Every GNFP-formula can be converted into this form in a canonical way with an exponential blow-up in size. We denote by GNFP^k the set of GNFP-formulas that are of width k when they are brought into this normal form. For convenience in proofs, we are using a slightly different normal form than previous papers on these logics.

These guarded fixpoint logics are expressive: the μ-calculus is contained in each of these fixpoint logics, and every positive existential formula is expressible in UNFP and GNFP (and even UNF and GNF). Nevertheless, these logics are decidable and have nice model theoretic properties. In particular satisfiability and finite satisfiability is 2-ExpTime-complete for GNF and GNFP [5]. The same holds for UNFP and GFP [23, 18]. GNFP (and hence UNFP and GFP) has the tree-like model property [5]: if ϕ is satisfiable, then ϕ is satisfiable over structures of bounded tree-width. In fact satisfiable GNFP^k formulas have satisfying structures of tree-width $k-1$. GNF (and hence UNF and GF) has the finite-model property [5]: if ϕ is satisfiable, then ϕ is satisfiable in a finite structure. This does not hold for the fixpoint extensions. In this paper we will be concerned with equivalence over all structures.

In this work we will be interested in varying the signatures considered, and in distinguishing more finely which relations can be used in guards. If we want to emphasize the relational signature σ being used, then we will write, e.g., $\text{GNFP}[\sigma]$. For $\sigma_g \subseteq \sigma$, we let $\text{GNFP}[\sigma, \sigma_g]$ denote the logic built up as in GNFP but allowing only relations $R \in \sigma$ at the atomic step and only guards α using equality or relations $R \in \sigma_g$. We define $\text{GFP}[\sigma, \sigma_g]$ similarly. Note that $\text{UNFP}[\sigma]$ is equivalent to $\text{GNFP}[\sigma, \emptyset]$, since if the only guards are equality guards, then the formula can be rewritten to use only unary negation and monadic fixpoints.

Guarded second-order logic over a signature σ (denoted $\text{GSO}[\sigma]$) is a fragment of second-order logic in which second-order quantification is interpreted only over guarded relations, i.e. over relations where every tuple in the relation is guarded by some predicate from σ. We refer the interested reader to [16] for more background and some equivalent definitions of this logic. The logics UNFP, GNFP, and GFP can all be translated into GSO.

A special kind of signature is a *transition system signature* Σ consisting of a finite set of unary predicates (corresponding to a set of propositions) and binary predicates (corresponding to a set of actions). A structure for such a signature is a *transition system*. Trees allowing both edge-labels and node-labels have a natural interpretation as transition systems. We will be interested in two logics over transition system signatures. One is *monadic second-order logic* (denoted MSO) – where second-order quantification is only over unary relations. MSO is contained in GSO, because unary relations are trivially guarded. While MSO and GSO can be interpreted over arbitrary signatures, there are logics like *modal logic* that have syntax specific to transition system signatures. Another is the *modal μ-calculus* (denoted L_μ), an extension of modal logic with fixpoints. Given a transition system signature Σ, formulas $\phi \in \text{L}_\mu[\Sigma]$ can be generated using the grammar $\phi ::= P \mid X \mid \phi \wedge \phi \mid \neg\phi \mid \langle\rho\rangle\phi \mid \mu X.\phi$ where P is a unary relation in Σ and ρ is a binary relation in Σ. The formulas $\mu X.\phi$ are required to use the variable X only positively in ϕ, and the semantics define a least-fixpoint operation based on ϕ. It is easy to see that L_μ can be translated into MSO.

It is well-known that σ-structures of tree-width $k-1$ can be encoded by labelled trees over an alphabet that depends only on the signature of the structure and k, which we denote $\Sigma^{\text{code}}_{\sigma,k}$. Our encoding scheme will make use of trees with both node and edge labels, i.e. trees over a

transition system signature $\Sigma^{\text{code}}_{\sigma,k}$. Roughly speaking, a node label is a set of unary relations (like $R_{i_1,\ldots,i_n}$) from $\Sigma^{\text{code}}_{\sigma,k}$ that encodes the set of atomic formulas (like $R(a_{i_1},\ldots,a_{i_n})$) that hold of the elements represented at that node, and an edge label ρ is a binary relation from $\Sigma^{\text{code}}_{\sigma,k}$ that indicates the relationship between the names of encoded elements in neighboring nodes. This scheme differs slightly from the one used in [16]. The exact coding conventions are not important for understanding the ideas in the rest of the paper. Given some $\Sigma^{\text{code}}_{\sigma,k}$-tree $\mathcal{T}$, we say $\mathcal{T}$ is *consistent* if it satisfies certain natural conditions that ensure that the tree actually corresponds to a code of some tree decomposition of a σ-structure. A consistent $\Sigma^{\text{code}}_{\sigma,k}$-tree $\mathcal{T}$ can be decoded to an actual σ-structure, denoted $\mathfrak{D}(\mathcal{T})$.

Bisimulation games and unravellings. The logic L_μ over transition system signatures lies within MSO. Similarly the guarded logics GFP, UNFP, and GNFP all lie within GSO and apply to arbitrary-arity signatures. It is easy to see that these containments are proper. In each case, what distinguishes the smaller logic from the larger is *invariance* under certain equivalences called *bisimulations*, each of which is defined by a certain player having a winning strategy in a two-player infinite game played between players Spoiler and Duplicator.

For L_μ, the appropriate game is the classical *bisimulation game* between transition systems $\mathfrak{A}$ and $\mathfrak{B}$. It is straightforward to check that $\mathrm{L}_\mu[\Sigma]$-formulas are Σ-bisimulation invariant, i.e. they cannot distinguish between Σ-bisimilar transition systems. We will make use of a stronger result of Janin and Walukiewicz [20] that the μ-calculus is the bisimulation-invariant fragment of MSO (we state it here for trees because of how we use this later): A class of trees is definable in $\mathrm{L}_\mu[\Sigma]$ iff it is definable in MSO$[\Sigma]$ and closed under Σ-bisimulation within the class of all Σ-trees. Moreover, the translation between these logics is effective.

We now describe a generalization of these games between structures $\mathfrak{A}$ and $\mathfrak{B}$ over a signature σ with arbitrary arity relations, parameterized by some subsignature σ' of the structures. Each position in the game is a partial σ' homomorphism h from $\mathfrak{A}$ to $\mathfrak{B}$, or vice versa. The *active structure* in position h is the structure containing the domain of h. The game starts from the empty partial map from $\mathfrak{A}$ to $\mathfrak{B}$. In each round of the game, Spoiler chooses between one of the following moves:

- *Extend:* Spoiler chooses some set X of elements in the active structure such that $X \supseteq \mathrm{dom}(h)$, and Duplicator must then choose h' extending h (i.e. such that $h(c) = h'(c)$ for all $c \in \mathrm{dom}(h)$) such that h' is a partial σ' homomorphism; Duplicator loses if this is not possible. Otherwise, the game proceeds from the position h'.
- *Switch:* Spoiler chooses to switch active structure. If h is not a partial σ' isomorphism, then Duplicator loses. Otherwise, the game proceeds from the position h^{-1}.
- *Collapse:* Spoiler selects some $X \subseteq \mathrm{dom}(h)$ and the game continues from position $h \restriction_X$.

Duplicator wins if she can continue to play indefinitely.

We will consider several variants of this game. These were essentially known already in the literature (see, e.g., [16, 17, 3]), sometimes with different names or minor technical differences in the definitions. For $k \in \mathbb{N}$ and $\sigma_g \subseteq \sigma'$:

1. **k-width guarded negation bisimulation game**: The $\mathrm{GN}^k[\sigma', \sigma_g]$-game is the version of the game where the domain of every position h is of size at most k, and Spoiler can only make a switch move at h if $\mathrm{dom}(h)$ is strictly σ_g-guarded in the active structure.
2. **block k-width guarded negation bisimulation game**: The $\mathrm{BGN}^k[\sigma', \sigma_g]$-game is like the $\mathrm{GN}^k[\sigma', \sigma_g]$-game, but additionally Spoiler is required to alternate between extend/switch moves and moves where he collapses to a strictly σ_g-guarded set. We call it the "block" game since Spoiler must select all of the new extension elements in a single block, rather than as a series of small extensions. The key property is that the game alternates

between positions with a strictly σ_g-guarded domain, and positions of size at most k. The restriction mimics the alternation between formulas of width at most k and strictly σ_g-guarded formulas within normalized GNFP^k formulas.

3. **guarded bisimulation game**: The $\mathrm{G}[\sigma', \sigma_g]$-game is the version of the game where the domain of every position must be strictly σ_g-guarded in the active structure. Note that in such a game, every position h satisfies $|\mathrm{dom}(h)| \leq \mathrm{width}(\sigma_g)$.

We say $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathrm{GN}^k[\sigma', \sigma_g]$-*bisimilar* if Duplicator has a winning strategy in the $\mathrm{GN}^k[\sigma', \sigma_g]$-game starting from the empty position. We say a sentence ϕ is $\mathrm{GN}^k[\sigma', \sigma_g]$-*invariant* if for any pair of $\mathrm{GN}^k[\sigma', \sigma_g]$-bisimilar σ'-structures, $\mathfrak{A} \models \phi$ iff $\mathfrak{B} \models \phi$. A logic $\mathcal{L}$ is $\mathrm{GN}^k[\sigma', \sigma_g]$-invariant if every sentence in $\mathcal{L}$ is $\mathrm{GN}^k[\sigma', \sigma_g]$-invariant. When the guard signature is the entire signature, we will write, e.g., $\mathrm{GN}^k[\sigma']$ instead of $\mathrm{GN}^k[\sigma', \sigma']$.

It is known that the bisimulation games characterize certain fragments of FO: $\mathrm{GF}[\sigma']$ is the $\mathrm{G}[\sigma']$-invariant fragment of $\mathrm{FO}[\sigma']$ [1] and $\mathrm{GNF}^k[\sigma']$ can be characterized as either the $\mathrm{BGN}^k[\sigma']$-invariant or the $\mathrm{GN}^k[\sigma']$-invariant fragment of $\mathrm{FO}[\sigma']$ (this follows from work in [3] and [7]). Likewise, for fixpoint logics and fragments of GSO, $\mathrm{GFP}[\sigma']$ is the $\mathrm{G}[\sigma']$-invariant fragment of $\mathrm{GSO}[\sigma']$ [16], while $\mathrm{UNFP}^k[\sigma']$ is the $\mathrm{BGN}^k[\sigma', \emptyset]$-invariant fragment of $\mathrm{GSO}[\sigma']$ (this follows from [9]). The survey in [17] also describes some of these invariance results.

In this paper, we will prove a corresponding characterization for $\mathrm{GNFP}^k[\sigma']$ in terms of $\mathrm{BGN}^k[\sigma']$-invariance: $\mathrm{GNFP}^k[\sigma']$ is the $\mathrm{BGN}^k[\sigma']$-invariant fragment of $\mathrm{GSO}[\sigma']$ (see Theorem 16). Note that for fixpoint logics, $\mathrm{GN}^k[\sigma']$-invariance is strictly weaker than $\mathrm{BGN}^k[\sigma']$-invariance, and applies to other decidable logics (e.g. [7]).

Unravellings. Given a σ-structure $\mathfrak{A}$ and $k \in \mathbb{N}$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$, we would like to construct a structure that is $\mathrm{GN}^k[\sigma', \sigma_g]$-bisimilar to $\mathfrak{A}$ but has a tree-decomposition of bounded tree-width. A standard construction achieves this, called the $\mathrm{GN}^k[\sigma', \sigma_g]$-unravelling of $\mathfrak{A}$. Let Π_k be the set of finite sequences of the form $Y_0 Y_1 \dots Y_m$ such that $Y_0 = \emptyset$ and each Y_i is a set of elements from $\mathfrak{A}$ of size at most k. Each such sequence can be seen as the projection to $\mathfrak{A}$ of a play in the $\mathrm{GN}^k[\sigma', \sigma_g]$-bisimulation game between $\mathfrak{A}$ and some other structure. For Y a set of elements from $\mathfrak{A}$, let $\mathrm{AT}_{\mathfrak{A},\sigma'}(Y)$ be the set of atoms that hold of the elements in Y: $\{R(a_1, \dots, a_l) : R \in \sigma', \{a_1, \dots, a_l\} \subseteq Y, \mathfrak{A} \models R(a_1, \dots, a_l)\}$. Now define a $\Sigma^{\mathrm{code}}_{\sigma',k}$-tree $\mathcal{U}_{\mathrm{GN}^k[\sigma',\sigma_g]}(\mathfrak{A})$ where each node corresponds to a sequence in Π_k, and the sequences are arranged in prefix order. Roughly speaking, the node label of every $v = Y_0 \dots Y_{m-1} Y_m$ is an encoding of $\mathrm{AT}_{\mathfrak{A},\sigma'}(Y_m)$, and the edge label between its parent u and v indicates the relationship between the shared elements $Y_{m-1} \cap Y_m$ encoded in u and v. We define $\mathfrak{D}(\mathcal{U}_{\mathrm{GN}^k[\sigma',\sigma_g]}(\mathfrak{A}))$ to be the $\mathrm{GN}^k[\sigma', \sigma_g]$-unravelling of $\mathfrak{A}$. By restricting the set Π_k to reflect the possible moves in the games, we can define unravellings based on the other bisimulation games in a similar fashion. We summarize the two unravellings that will be most relevant:

1. **block k-width guarded negation unravelling**: The $\mathrm{BGN}^k[\sigma', \sigma_g]$-unravelling is denoted $\mathfrak{D}(\mathcal{U}_{\mathrm{BGN}^k[\sigma',\sigma_g]}(\mathfrak{A}))$. Its encoding $\mathcal{U}_{\mathrm{BGN}^k[\sigma',\sigma_g]}(\mathfrak{A})$ is obtained by considering only sequences $Y_0 \dots Y_m \in \Pi_k$ such that for all *even* i, $Y_{i-1} \supseteq Y_i$ and $Y_i \subseteq Y_{i+1}$ and Y_i is strictly σ_g-guarded in $\mathfrak{A}$. The tree $\mathcal{U}_{\mathrm{BGN}^k[\sigma',\sigma_g]}(\mathfrak{A})$ is consistent and is called a *σ_g-guarded-interface tree* since it alternates between *interface nodes* with strictly σ_g-guarded domains – corresponding to collapse moves in the game – and *bag nodes* with domain of size at most k that are not necessarily σ_g-guarded.
2. **guarded unravelling**: The $\mathrm{G}[\sigma', \sigma_g]$-unravelling is denoted $\mathfrak{D}(\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{A}))$ and its encoding $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{A})$ is obtained by considering only sequences $Y_0 \dots Y_m \in \Pi_k$ such that for all i, Y_i is strictly σ_g-guarded in $\mathfrak{A}$. The tree $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{A})$ is consistent and is called a *σ_g-guarded* tree since the domain of every node in the tree is strictly σ_g-guarded.

All of these unravellings are bisimilar to $\mathfrak{A}$, with respect to the appropriate notion of bisimilarity. Because these unravellings have tree decompositions of some bounded tree-width, this proposition implies that these guarded logics have tree-like models. The structural differences in the tree decompositions will be exploited for our definability decision procedures.

3 Decidability via back-and-forth and equivalence

We now give the main components of our approach, and explain how they fit together.

The first component is a *forward mapping*, translating an input GSO formula ϕ_0 to an MSO formula ϕ_0' over tree-codes, holding on the codes that correspond to tree-like models of ϕ_0. We will be interested only in formulas that are invariant under a form of guarded bisimulation or guarded negation bisimulation, so we assume the input is a GN^l-invariant formula, for some $l \geq \mathrm{width}(\sigma)$. For such formulas, we can actually define a forward mapping that produces a μ-calculus formula.

▶ **Lemma 1** (Fwd, adapted from [16]). *Given a $\mathrm{GN}^l[\sigma]$-invariant sentence $\phi \in \mathrm{GSO}[\sigma]$ and given some $k \geq \mathrm{width}(\sigma)$, we can construct $\phi^\mu \in \mathrm{L}_\mu[\Sigma^{code}_{\sigma,\max\{k,l\}}]$ such that for all consistent $\Sigma^{code}_{\sigma,\max\{k,l\}}$-trees $\mathcal{T}$, $\mathcal{T} \models \phi^\mu$ iff $\mathfrak{D}(\mathcal{T}) \models \phi$.*

The second component will depend on our target sublogic $\mathcal{L}_1$. It requires a mapping (not necessarily effective) taking a σ-structure $\mathfrak{B}$ to a tree structure $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ such that $\mathfrak{D}(\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B}))$ agrees with $\mathfrak{B}$ on all $\mathcal{L}_1$ sentences. Informally, $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ will be the encoding of some unravelling of $\mathfrak{B}$ appropriate for $\mathcal{L}_1$, perhaps with additional properties. A *backward mapping for* $\mathcal{L}_1$ takes sentences ϕ_0' over tree codes (with some given k and σ) to a sentence $\phi_1 \in \mathcal{L}_1$ such that: for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \phi_1$ iff $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B}) \models \phi_0'$.

The formula ϕ_1 will depend on simplifying the formula ϕ_0' based on the fact that one is working on an unravelling. For $\mathcal{L}_1 = \mathrm{GFP}[\sigma', \sigma_g]$ over subsignatures σ', σ_g of the original signature σ, $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ will be a guarded unravelling; the results of [16] can easily be refined to give the formula component in $\mathrm{GFP}[\sigma', \sigma_g]$. For GNFP^k, providing both the appropriate unravelling and the formula in the backward mappings will require more work.

The $\mathcal{L}_1$ *definability problem for logic* $\mathcal{L}$ asks: given some input sentence $\phi \in \mathcal{L}$, is there some $\psi \in \mathcal{L}_1$ such that ϕ and ψ are logically equivalent? The forward and backward method of Figure 1b gives us a generic approach to this problem. The algorithm consists of applying the forward mapping to get ϕ_0', applying the backward mapping to ϕ_0' based on $\mathcal{L}_1$ to get ϕ_1, and then checking if ϕ_1 is equivalent to ϕ_0. We claim ϕ_0 is $\mathcal{L}_1$ definable iff ϕ_0 and ϕ_1 are equivalent. If ϕ_0 and ϕ_1 are logically equivalent then ϕ_0 is clearly $\mathcal{L}_1$ definable using ϕ_1. In the other direction, suppose that ϕ_0 is $\mathcal{L}_1$-definable. Fix $\mathfrak{B}$, and let $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$ be given by the backward mapping. Then

$$\begin{aligned}\mathfrak{B} \models \phi_0 &\Leftrightarrow \mathfrak{D}(\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})) \models \phi_0 \text{ since } \mathfrak{D}(\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})) \text{ agrees with } \mathfrak{B} \text{ on } \mathcal{L}_1 \text{ sentences} \\ &\Leftrightarrow \mathcal{U}_{\mathcal{L}_1}(\mathfrak{B}) \models \phi_0' \text{ by Lemma Fwd} \Leftrightarrow \mathfrak{B} \models \phi_1 \text{ by Backward Mapping for } \mathcal{L}_1.\end{aligned}$$

Hence, ϕ_0 and ϕ_1 are logically equivalent, as required. Thus, we get the following general decidability result:

▶ **Proposition 2.** *Let $\mathcal{L}_1$ be a subset of $\mathrm{GN}^l[\sigma]$-invariant $\mathrm{GSO}[\sigma]$ such that we have an effective backward mapping for $\mathcal{L}_1$. Then the $\mathcal{L}_1$ definability problem is decidable for $\mathrm{GN}^l[\sigma]$-invariant $\mathrm{GSO}[\sigma]$.*

Above, we mean that there is an algorithm that decides $\mathcal{L}_1$ definability for any input $\mathrm{GSO}[\sigma]$ sentence that is $\mathrm{GN}^l[\sigma]$-invariant, with the output being arbitrary otherwise. The

approach above gives a definability test in the usual sense for inputs in GNFP[σ], since these are all GN^l[σ]-invariant for some l. In particular we will see that we can test whether a GNFP^l[σ] sentence is in GFP[σ'] or in GNFP^k[σ']. But there are larger GN^l-invariant logics (e.g. [7]), and the algorithm immediately applies to these as well.

4 Identifying GFP definable sentences

For GFP, we can instantiate the high-level algorithm by giving a backward mapping.

▶ **Lemma 3** (GFP-Bwd, adapted from [16]). *Given $\phi^\mu \in \mathrm{L}_\mu[\Sigma^{code}_{\sigma,m}]$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$, ϕ^μ can be translated into $\psi \in \mathrm{GFP}[\sigma', \sigma_g]$ such that for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \psi$ iff $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{B}) \models \phi^\mu$.*

Plugging this into our high-level algorithm, with $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{B})$ as $\mathcal{U}_{\mathcal{L}_1}(\mathfrak{B})$, we get decidability of the GFP-definability problem:

▶ **Theorem 4.** *The* GFP[σ', σ_g] *definability problem is decidable for* GN^k[σ]*-invariant* GSO[σ] *where $k \geq \mathrm{width}(\sigma)$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$.*

There are two sources of inefficiency in the high-level algorithm. First, the forward mapping is non-elementary since we pass through MSO on the way to a μ-calculus formula. Second, testing equivalence of the original sentence with the sentence produced by the forward and backward mappings implicitly requires a second forward mapping in order to reduce the problem to regular language equivalence on trees.

For the special case of input in GNFP, we can use an optimized procedure that avoids these inefficiencies and allows us to obtain an optimal complexity bound.

▶ **Theorem 5.** *The* GFP[σ', σ_g] *definability problem is* 2-ExpTime*-complete for input in* GNFP[σ].

The main idea behind our optimized procedure is to directly use automata throughout the process. First, for input ϕ in GNFP it is known from [9] how to give a forward mapping that directly produces a tree automaton $\mathcal{A}_\phi$ (with exponentially-many states) that accepts a consistent tree $\mathcal{T}$ iff $\mathfrak{D}(\mathcal{T}) \models \phi$ – exactly the consistent trees that satisfy ϕ^μ. This direct construction avoids passing through MSO, and can be done in 2-ExpTime. We can then construct an automaton $\mathcal{A}'_\phi$ from $\mathcal{A}_\phi$ that accepts a tree $\mathcal{T}$ iff $\mathcal{U}_{\mathrm{G}[\sigma',\sigma_g]}(\mathfrak{D}(\mathcal{T}))$; we call this the G[$\sigma', \sigma_g$]*-view automaton*, since it mimics the view of $\mathcal{T}$ running on the guarded unravelling of $\mathfrak{D}(\mathcal{T})$. This can be seen as an automaton that represents the composition of the backward mapping with the forward mapping. With these constructions in place, we have the following improved algorithm to test definability of ϕ in GFP: construct $\mathcal{A}_\phi$ from ϕ, construct $\mathcal{A}'_\phi$ from $\mathcal{A}_\phi$, and test equivalence of $\mathcal{A}_\phi$ and $\mathcal{A}'_\phi$ over consistent trees. Note that with this improved procedure it is not necessary to actually construct the backward mapping, or to pass forward to trees for a second time in order to test equivalence. Overall, the procedure can be shown to run in 2-ExpTime. A reduction from GFP-satisfiability testing, which is known to be 2-ExpTime-hard, yields the lower bound.

Our results give us a corollary on definability in fragments of FO when the input is in FO:

▶ **Corollary 6.** *The* GF[σ', σ_g] *definability problem is decidable for* GN^l[σ]*-invariant* FO[σ] *where $k, l \geq \mathrm{width}(\sigma)$ and $\sigma_g \subseteq \sigma' \subseteq \sigma$.*

Note that in this work we are characterizing sublogics within fragments of fixpoint logics and within fragments of first-order logic. We do not deal with identifying first-order definable formulas within a fixpoint logic, as in [10, 8].

We also can get a version of the definability result for a restriction of fixpoint logic. One well-studied restriction is called alternation-freeness (see, e.g., [14, 2]). We say a sentence ϕ in GFP is *alternation-free* if it does not contain subformulas $\psi_1 := [\mathbf{lfp}_{Y,\boldsymbol{y}} . \chi_1](\boldsymbol{y})$ and $\psi_2 := [\mathbf{gfp}_{Z,\boldsymbol{z}} . \chi_2](\boldsymbol{z})$ such that Y occurs in χ_2 and ψ_2 is a subformula of ψ_1, or Z occurs in χ_1 and ψ_1 is a subformula of ψ_2 (recall that a greatest fixpoint can be defined in terms of least fixpoints and negations as $[\mathbf{gfp}_{Z,\boldsymbol{z}} . \chi_2](\boldsymbol{z}) \equiv \neg[\mathbf{lfp}_{Z,\boldsymbol{z}} . \neg\chi_2[\neg Z/Z]](\boldsymbol{z})$). Alternation-free fragments of GNFP and L_μ are defined by restricting the nesting of fixpoints in a similar way. It is desirable to know if a sentence is in this alternation-free fragment of GFP since this fragment has better computational properties: for instance, model checking for this alternation-free fragment can be done in linear time. This was shown in [14]. A language called DATALOG-LITE – a variant of DATALOG that has some restricted forms of negation and universal quantification – was also introduced, and shown to exactly characterize this alternation-free GFP [14]. A corollary of the definability result in this section, is that it is possible to decide definability in alternation-free GFP when the input is in alternation-free GNFP, using the same decision procedure as before. Roughly speaking, this comes from observing that if the input is in alternation-free GNFP, then the forward mapping produces alternation-free L_μ, and the backward mapping produces alternation-free GFP.

▶ **Corollary 7.** *The alternation-free* GFP$[\sigma']$ *(equivalently,* DATALOG-LITE$[\sigma']$*) definability problem is decidable in* 2-ExpTime *for input in alternation-free* GNFP$[\sigma]$.

We can also apply our theorem to answer some questions about *conjunctive queries (CQs)*: formulas built up from relational atoms via $\wedge$ and $\exists$. When the input ϕ to our definability algorithm is a CQ, ϕ can be written as a GF sentence exactly when it is *acyclic*: roughly speaking, this means it can be built up from guarded existential quantification (see [15]). Transforming a query to an acyclic one could be quite relevant in practice, since acyclic queries can be evaluated in linear time [26]. There are well-known methods for deciding whether a CQ ϕ is acyclic, and recently these have been extended to the problem of determining whether ϕ is acyclic for all structures satisfying a set of constraints (e.g., Guarded TGDs [6] or Functional Dependencies [13]). Using Corollary 6 above along with an equivalence between guardedness and acyclicity that follows from [4], we can get an analogous result for arbitrary constraints in the guarded fragment:

▶ **Corollary 8.** *Given a set of* GF *sentences* Σ *and a CQ sentence* Q*, we can decide whether there is a union of acyclic CQs* Q' *equivalent to* Q *for all structures satisfying* Σ*. The problem is* 2-ExpTime*-complete.*

Note that if Σ consists of universal Horn constraints ("TGDs"), then a CQ Q is equivalent to union of CQs Q' relative to Σ implies that it is equivalent to one of the disjuncts of Q'. Thus the result above implies decidability of acyclicity relative to universal horn GF sentences, one of the main results of [6].

5 Identifying GNFP^k and UNFP^k sentences

We now turn to extending the prior results to GNFP and UNFP. In order to make use of the back-and-forth approach described in the previous section, we must be able to restrict to structures of some bounded tree-width. For GFP this tree-width depends only on the signature σ', so this width-restriction was implicit in the GFP$[\sigma', \sigma_g]$-definability problems. However, for GNFP and UNFP this bound on the tree-width depends on the width of the formula, so for definability questions, we must state this width explicitly. Hence, in this section, we consider definability questions related to GNFP^k and UNFP^k.

We apply the high-level algorithm of Proposition 2, using the forward mapping of Lemma 1. The unravelling and backward mapping for GNFP^k is more technically challenging than the corresponding construction for GFP.

We first need an appropriate notion of unravelling. We use a variant of the block k-width guarded negation unravelling discussed in Section 2, but we will need to assume we have a certain repetition of facts. This idea of modifying a classical unravelling to include extra copies of certain pieces of the structure has been used before (e.g. the ω-expansions in [21], and "shrewd" unravellings for UNFP^k in [9]). We will need a new, subtler property for GNFP^k, which we call "plumpness".

In order to define the property that this special unravelling has, we need to define how we can modify copies of certain parts of the structure in a way that still leads to a GN^k-bisimilar structure. Let τ and τ' be sets of σ'-facts over some elements A. Let $I, J \subseteq A$. We say τ and τ' agree on J if for all σ'-atoms $R(a_1, \ldots, a_l)$ with $\{a_1, \ldots, a_l\} \subseteq J$, $R(a_1, \ldots, a_l) \in \tau$ iff $R(a_1, \ldots, a_l) \in \tau'$. We say τ' is an (σ_g, I)-*safe restriction* of τ if (i) $\tau' \subseteq \tau$; (ii) τ' agrees with τ on I; (iii) τ' agrees with τ on every $J \subseteq A$ that is σ_g-guarded in τ'. Note that τ itself is considered a trivial (σ_g, I)-safe restriction of τ. Here is another example:

▶ **Example 9.** Consider signatures $\sigma' = \{U, R, T\}$ and $\sigma_g = \{R\}$, where U is a unary relation, R is a binary relation, and T is a ternary relation. Consider $I = \{1, 2\}$ and $\tau = \{U(1),\, U(3),\, R(1,2),\, R(2,3),\, R(3,1),\, T(3,2,2)\}$. Then the possible (σ_g, I)-safe restrictions of τ are τ itself and

$$\tau'_1 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2),\, R(2,3) \\ T(3,2,2) \end{array} \right\} \qquad \tau'_2 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2),\, R(3,1) \\ T(3,2,2) \end{array} \right\} \qquad \tau'_4 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2) \\ T(3,2,2) \end{array} \right\}$$

$$\tau'_3 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2),\, R(3,1) \end{array} \right\} \qquad \tau'_5 = \left\{ \begin{array}{c} U(1),\, U(3) \\ R(1,2) \end{array} \right\}.$$

Note that we cannot drop facts over unary relations (since these are always trivially guarded), and we can never drop facts over I. Further, the σ_g-facts that we keep restrict what other facts we can drop, since for any σ_g-guarded set that remains we must preserve facts over that set. By a (σ_g, I)-safe restriction of a node in a tree decomposition, we mean a (σ_g, I)-safe restriction of the atoms represented by the node. We will be interested in trees with the property that for every bag node w, all safe restrictions of w are realized by siblings of w. Formally a $\Sigma^{\mathrm{code}}_{\sigma',k}$-tree has the σ_g-*plumpness property* if for all interface nodes v: if w is a ρ_0-child of v over names J with $I = \mathrm{rng}(\rho_0)$ and τ is the encoded set of σ'-atoms that hold at w, then for any (σ_g, I)-safe restriction τ' of τ, there is a ρ_0-child w' of v such that (i) τ' is the encoded set of σ'-atoms that hold at w'; (ii) for each ρ-child u' of w', there is a ρ-child u of w such that the subtrees rooted at u and u' are bisimilar; and (iii) for each ρ-child u of w such that $\mathrm{dom}(\rho)$ is strictly σ_g-guarded in τ', there is a ρ-child u' of w' such that the subtrees rooted at u' and u are bisimilar.

▶ **Example 10.** Let $\mathcal{T}$ be a plump tree. Suppose there is an interface node v in $\mathcal{T}$ with label encoding $\tau_0 = \{U(1), R(1,2)\}$, and there is a ρ_0-child w of v such that the label of w is the encoding of $\tau = \{U(1), U(3), R(1,2), R(2,3), R(3,1), T(3,2,2)\}$ from Example 9, and ρ_0 is the identity function with domain $\{1, 2\}$. Then by plumpness there must also be ρ_0-children $w_1, \ldots, w_5$ of v with labels encoding $\tau'_1, \ldots, \tau'_5$ from Example 9.

The following proposition shows that one can obtain unravellings that are plump:

▶ **Proposition 11.** *Let $\mathfrak{B}$ be a σ-structure, $k \in \mathbb{N}$, and $\sigma_g \subseteq \sigma' \subseteq \sigma$. There is a consistent, plump, σ_g-guarded-interface tree $\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B})$ such that $\mathfrak{B}$ is $\text{BGN}^k[\sigma',\sigma_g]$-bisimilar to $\mathfrak{D}(\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}))$. We call $\mathfrak{D}(\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}))$ the* plump unravelling *of $\mathfrak{B}$.*

Returning to the components required for the application of Proposition 2, we see that Proposition 11 says that $\mathfrak{B}$ is $\text{BGN}^k[\sigma',\sigma_g]$-bisimilar to $\mathfrak{D}(\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}))$ as required for an application of Proposition 2. Plumpness will come into play in the backward mapping:

▶ **Lemma 12** (GNFP^k-Bwd). *Given $\phi^\mu \in \text{L}_\mu[\Sigma^{code}_{\sigma,m}]$, relational signatures σ_g and σ' with $\sigma_g \subseteq \sigma' \subseteq \sigma$, and $k \leq m$, we can construct $\psi \in \text{GNFP}^k[\sigma',\sigma_g]$ such that for all σ-structures $\mathfrak{B}$, $\mathfrak{B} \models \psi$ iff $\mathcal{U}^{\text{plump}}_{\text{BGN}^k[\sigma',\sigma_g]}(\mathfrak{B}) \models \phi^\mu$.*

There is a naïve backward mapping of the μ-calculus into LFP, by structural induction. The problem is that the formula produced by the translation fails to be in GNFP^k for two reasons. First, the inductive step for negation in the naïve algorithm simply applies negation to the recursively-produced formula. Clearly this can produce unguarded negation. Similarly, the recursive step for fixpoints may use unguarded fixpoints.

For example, the original μ-calculus formula can include subformulas of the form $\langle\rho\rangle\text{ExactLabel}(\tau)$ where τ is a set of unary relations from $\Sigma^{\text{code}}_{\sigma',k}$, and $\text{ExactLabel}(\tau)$ asserts P for all $P \in \tau$ and $\neg P$ for all unary relations P not in τ. This would be problematic for a straightforward backward mapping, since the backward translation of some $\neg R_{i_1,\ldots,i_n}$ would be converted into an unguarded negation $\neg R(x_{i_1},\ldots,x_{i_n})$. On the other hand the formula $\langle\rho\rangle\text{GNLabel}(\tau)$ where $\text{GNLabel}(\tau)$ asserts P for all $P \in \tau$ but only asserts $\neg P$ for unary relations P that are not in τ but whose indices are σ_g-guarded by some $P' \in \tau$ would be unproblematic, since this could be translated to a formula with σ_g-guarded negation. The key observation is that from an interface node in a plump tree, these two formulas are equivalent: if $\mathcal{T}, v \models \langle\rho\rangle\text{GNLabel}(\tau)$ at any interface node v, then plumpness ensures that if there is some ρ-child w' of v with label τ' satisfying $\text{GNLabel}(\tau)$, then there is a ρ-child w of v with label τ satisfying $\text{ExactLabel}(\tau)$ – it can be checked that τ is a $(\sigma_g, \text{rng}(\rho))$-safe restriction of τ'. Thus the proof of Lemma GNFP^k-Bwd relies on first simplifying L_μ-formulas so that problematic subformulas like $\text{ExactLabel}(\tau)$ are eliminated, with the correctness of this simplification holding only over plump trees. After this simplification, an inductive backward mapping can be applied.

Using the above lemma and Proposition 2, we obtain the following analog of Theorem 4.

▶ **Theorem 13.** *The $\text{GNFP}^k[\sigma',\sigma_g]$ definability problem is decidable for $\text{GN}^l[\sigma]$-invariant $\text{GSO}[\sigma]$ and $k, l \geq \text{width}(\sigma)$.*

Since $\text{UNFP}^k[\sigma']$ is just $\text{GNFP}^k[\sigma',\emptyset]$, we obtain the following corollary:

▶ **Corollary 14.** *The $\text{UNFP}^k[\sigma']$ definability problem is decidable for $\text{GN}^l[\sigma]$-invariant $\text{GSO}[\sigma]$ and $k, l \geq \text{width}(\sigma)$.*

We get corollaries for fragments of FO, analogous to Corollary 6:

▶ **Corollary 15.** *The $\text{GNF}^k[\sigma',\sigma_g]$ and $\text{UNF}^k[\sigma']$ definability problems are decidable for $\text{GN}^l[\sigma]$-invariant $\text{FO}[\sigma]$ and $k, l \geq \text{width}(\sigma)$.*

We can also apply the backward and forward mappings to get a semantic characterization for GNFP^k, analogous to the Janin-Walukiewicz theorem. The following extends a result of [3] characterizing GNF^k formulas as the BGN^k-invariant fragment of FO.

▶ **Theorem 16.** *$\text{GNFP}^k[\sigma',\sigma_g]$ is the $\text{BGN}^k[\sigma',\sigma_g]$-invariant fragment of $\text{GSO}[\sigma']$.*

The proof is similar to the characterizations of Janin-Walukiewicz and [16], and can also be seen as a variant of Proposition 2, where we use $\mathrm{BGN}^k[\sigma', \sigma_g]$-invariance rather than equivalence to a $\mathrm{GNFP}^k[\sigma', \sigma_g]$ sentence in justifying that the input formula is equivalent to the result of the composition of backward and forward mappings.

Interpolation. The forward and backward mappings utilized for the definability questions can also be used to prove that GFP and GNFP^k have a form of interpolation.

Let ϕ_{L} and ϕ_{R} be sentences over signatures σ_{L} and σ_{R} such that $\phi_{\mathrm{L}} \models \phi_{\mathrm{R}}$ (ϕ_{L} entails ϕ_{R}). An *interpolant* for such a validity is a formula θ for which $\phi_{\mathrm{L}} \models \theta$ and $\theta \models \phi_{\mathrm{R}}$, and θ mentions only relations appearing in both ϕ_{L} and ϕ_{R}. We say a logic $\mathcal{L}$ has *Craig interpolation* if for all $\phi_{\mathrm{L}}, \phi_{\mathrm{R}} \in \mathcal{L}$ with $\phi_{\mathrm{L}} \models \phi_{\mathrm{R}}$, there is an interpolant $\theta \in \mathcal{L}$ for it. We say a logic $\mathcal{L}$ has the stronger *uniform interpolation* property if one can obtain θ from ϕ_{L} and a signature σ', and θ can serve as an interpolant for any ϕ_{R} entailed by ϕ_{L} and such that the common signature of ϕ_{R} and ϕ_{L} is contained in σ'. A uniform interpolant can be thought of as the best over-approximation of ϕ_{L} over σ'.

Uniform interpolation holds for L_μ [11] and UNFP^k [9]. Unfortunately, $\mathrm{GFP}[\sigma]$ and $\mathrm{GNFP}^k[\sigma]$ both fail to have uniform interpolation and Craig interpolation (this follows from [19, 9]). However, if we disallow subsignature restrictions that change the guard signature, then we regain interpolation. This "preservation of guard" variant was investigated first by Hoogland, Marx, and Otto in the context of Craig interpolation [19]. The uniform variant was introduced by D'Agostino and Lenzi [12], who called it *uniform modal interpolation*. Formally, we say a guarded logic $\mathcal{L}[\sigma, \sigma_g]$ with guard signature $\sigma_g \subseteq \sigma$ has *uniform modal interpolation* if for any $\phi_{\mathrm{L}} \in \mathcal{L}[\sigma, \sigma_g]$ and any subsignature $\sigma' \subseteq \sigma$ containing σ_g, there exists a formula $\theta \in \mathcal{L}[\sigma', \sigma_g]$ such that ϕ_{L} entails θ and for any σ'' containing σ_g with $\sigma'' \cap \sigma \subseteq \sigma'$ and any $\phi_{\mathrm{R}} \in \mathcal{L}[\sigma'', \sigma_g]$ entailed by ϕ_{L}, θ entails ϕ_{R}. It was shown in [12] that GF has uniform modal interpolation. We strengthen this to GFP and GNFP^k.

▶ **Theorem 17.** *For σ a relational signature, $\sigma_g \subseteq \sigma$, $k \in \mathbb{N}$: $\mathrm{GFP}[\sigma, \sigma_g]$ and $\mathrm{GNFP}^k[\sigma, \sigma_g]$ sentences have uniform modal interpolation, and the interpolants can be found effectively.*

We sketch the argument for $\mathrm{GFP}[\sigma, \sigma_g]$. Consider $\phi_{\mathrm{L}} \in \mathrm{GFP}[\sigma, \sigma_g]$ of width k and subsignature $\sigma' \subseteq \sigma$ containing σ_g. We apply Lemma Fwd to get a formula $\phi_{\mathrm{L}}^\mu \in \mathrm{L}_\mu[\Sigma^{\mathrm{code}}_{\sigma,k}]$ that captures codes of tree-like models of ϕ_{L}. We want to go backward now, to get a formula over the subsignature σ'. We saw that the backward mapping for $\mathrm{GFP}[\sigma', \sigma_g]$ (Lemma 3) can do this: it can start with a μ-calculus formula over $\Sigma^{\mathrm{code}}_{\sigma,k}$, and produce a formula in $\mathrm{GFP}[\sigma', \sigma_g]$. As discussed earlier, the formula produced by this backward mapping has a nice property related to definability: it is equivalent to ϕ_{L} exactly when ϕ_{L} is definable in $\mathrm{GFP}[\sigma', \sigma_g]$. In general, however, we do not expect ϕ_{L} to be equivalent to a formula over the subsignature – for uniform interpolation we just want to *approximate* the formula over this subsignature. The backward mapping of ϕ_{L}^μ does not always do this. Hence, it is necessary to add one additional step before taking the backward mapping: we apply uniform interpolation for the μ-calculus [11], obtaining $\theta^\mu \in \Sigma^{\mathrm{code}}_{\sigma',k}$ which is entailed by ϕ_{L}^μ and entails each $\mathrm{L}_\mu[\Sigma^{\mathrm{code}}_{\sigma',k}]$-formula implied by ϕ_{L}^μ. Finally, we apply Lemma $\mathrm{GFP}[\sigma', \sigma_g]$-Bwd to θ^μ to get $\theta \in \mathrm{GFP}[\sigma', \sigma_g]$. We can check that $\theta \in \mathrm{GNFP}^k[\sigma', \sigma_g]$ is the required uniform modal interpolant for ϕ_{L} over subsignature σ'.

Theorem 17 also implies that UNFP^k has the traditional uniform interpolation property: since the guard signature is empty for UNFP^k, uniform modal interpolation and uniform interpolation coincide. This was shown already in [9].

6 Conclusions

In this paper we have taken a first look at effective characterizations of definability in expressive logics. We did not allow constants in the formulas in this paper, but we believe that similar effective characterization results hold for guarded fixpoint logics with constants. We leave open the question of definability in GNFP without any width restriction. For this the natural way to proceed is to bound the width of a defining sentence based in terms of the input. We also note that our results on fixpoint logics hold only when equivalence is considered over all structures, leaving open the corresponding questions over finite structures.

Acknowledgements. We thank the referees for many improvements.

References

1 Hajnal Andréka, Johan van Benthem, and István Németi. Modal languages and bounded fragments of predicate logic. *J. Phil. Logic*, 27:217–274, 1998.

2 Andre Arnold and Damin Niwinski. *Rudiments of mu-calculus*. Elsevier, 2001.

3 Vince Bárány, Balder Ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3), 2015. doi:10.1145/2701414.

4 Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. In *LMCS*, volume 10, 2014. doi:10.2168/LMCS-10(2:3)2014.

5 Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. In *ICALP*, 2011. doi:10.1007/978-3-642-22012-8_28.

6 Pablo Barceló, Georg Gottlob, and Andreas Pieris. Semantic acyclicity under constraints. In *PODS*, 2016. doi:10.1145/2902251.2902302.

7 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A step up in expressiveness of decidable fixpoint logics. In *LICS*, 2016. doi:10.1145/2933575.2933592.

8 Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *LICS*, 2015. doi:10.1109/LICS.2015.36.

9 Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Interpolation with decidable fixpoint logics. In *LICS*, 2015. doi:10.1109/LICS.2015.43.

10 Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. *Logical Methods in Computer Science*, 10(3), 2014. doi:10.2168/LMCS-10(3:2)2014.

11 Giovanna D'Agostino and Marco Hollenberg. Logical Questions Concerning The mu-Calculus: Interpolation, Lyndon and Los-Tarski. *The Journal of Symbolic Logic*, 65(1):310–332, 2000. doi:10.2307/2586539.

12 Giovanna D'Agostino and Giacomo Lenzi. Bisimulation quantifiers and uniform interpolation for guarded first order logic. *Theor. Comput. Sci.*, 563:75–85, 2015. doi:10.1016/j.tcs.2014.08.015.

13 Diego Figueira. Semantically acyclic conjunctive queries under functional dependencies. In *LICS*, 2016. doi:10.1145/2933575.2933580.

14 Georg Gottlob, Erich Grädel, and Helmut Veith. Datalog LITE: a deductive query language with linear time model checking. *ACM Trans. Comput. Log.*, 3(1):42–79, 2002. doi:10.1145/504077.504079.

15 Georg Gottlob, Nicola Leone, and Francesco Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *J. Comput. Syst. Sci.*, 66(4):775–808, 2003. doi:10.1016/S0022-0000(03)00030-8.

16 Erich Grädel, Colin Hirsch, and Martin Otto. Back and forth between guarded and modal logics. *ACM TOCL*, 3(3):418–463, 2002. doi:10.1145/507382.507388.

17 Erich Grädel and Martin Otto. The freedoms of (guarded) bisimulation. In *Johan van Benthem on Logic and Information Dynamics*, pages 3–31. Springer, 2014. doi:10.1007/978-3-319-06025-5_1.

18 Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *LICS*, 1999. doi:10.1109/LICS.1999.782585.

19 Eva Hoogland, Maarten Marx, and Martin Otto. Beth definability for the guarded fragment. In *LPAR*, 1999. doi:10.1007/3-540-48242-3_17.

20 David Janin and Igor Walukiewicz. Automata for the modal mu-calculus and related results. In *MFCS*, 1995. doi:10.1007/3-540-60246-1_160.

21 David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *CONCUR*, 1996. doi:10.1007/3-540-61604-7_60.

22 Martin Otto. Eliminating recursion in the μ-calculus. In *STACS*, 1999. doi:10.1007/3-540-49116-3_50.

23 Balder ten Cate and Luc Segoufin. Unary negation. In *STACS*, 2011. doi:10.4230/LIPIcs.STACS.2011.344.

24 J. F. A. K. van Benthem. *Modal Logic and Classical Logic*. Humanities Pr, 1983.

25 Moshe Y. Vardi. "Why is Modal Logic so Robustly Decidable". In *Descriptive Complexity and Finite Models*, pages 149–184, 1997.

26 Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, 1981.

Conservative Extensions in Guarded and Two-Variable Fragments*†

Jean Christoph Jung[1], Carsten Lutz[1], Mauricio Martel[1], Thomas Schneider[1], and Frank Wolter[2]

1 **University of Bremen, Bremen, Germany**
`jeanjung@informatik.uni-bremen.de`
2 **University of Bremen, Bremen, Germany**
`clu@informatik.uni-bremen.de`
3 **University of Bremen, Bremen, Germany**
`martel@informatik.uni-bremen.de`
4 **University of Bremen, Bremen, Germany**
`ts@informatik.uni-bremen.de`
5 **University of Liverpool, Liverpool, UK**
`wolter@liverpool.ac.uk`

Abstract

We investigate the decidability and computational complexity of (deductive) conservative extensions in fragments of first-order logic (FO), with a focus on the two-variable fragment FO^2 and the guarded fragment GF. We prove that conservative extensions are undecidable in any FO fragment that contains FO^2 or GF (even the three-variable fragment thereof), and that they are decidable and 2ExpTime-complete in the intersection GF^2 of FO^2 and GF.

1998 ACM Subject Classification F.4.1 [Mathematical Logic] Computational Logic

Keywords and phrases Conservative Extensions, Decidable Fragments of First-Order Logic, Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.108

1 Introduction

Conservative extensions are a fundamental notion in logic. In mathematical logic, they provide an important tool for relating logical theories, such as theories of arithmetic and theories that emerge in set theory [35, 31]. In computer science, they come up in diverse areas such as software specification [12], higher order theorem proving [15], and ontologies [24]. In these applications, it would be very useful to decide, given two sentences φ_1 and φ_2, whether $\varphi_1 \wedge \varphi_2$ is a conservative extension of φ_1. As expected, this problem is undecidable in first-order logic (FO). In contrast, it has been observed in recent years that conservative extensions are decidable in many modal and description logics [13, 26, 27, 7]. This observation is particularly interesting from the viewpoint of ontologies, where conservative extensions have many natural applications including modularity and reuse, refinement, versioning, and forgetting [9, 24].

Regarding decidability, conservative extensions thus seem to behave similarly to the classical satisfiability problem, which is also undecidable in FO while it is decidable for modal

* A full version of the paper is available at `https://arxiv.org/abs/1705.10115`.
† Funded by DFG grant LU 1417/2.

and description logics. In the case of satisfiability, the aim to understand the deeper reasons for this discrepancy and to push the limits of decidability to more expressive fragments of FO has sparked a long line of research that identified prominent decidable FO fragments such as the two-variable fragment FO^2 [34, 29], its extension with counting quantifiers C^2 [19], the guarded fragment GF [1], and the guarded negation fragment GNF [4], see also [6, 16, 33, 23] and references therein. These fragments have sometimes been used as a replacement for the modal and description logics that they generalize, and in particular the guarded fragment has been proposed as an ontology language [3]. Motivated by this situation, the aim of the current paper is to study the following two questions:

1. Are conservative extensions decidable in relevant fragments of FO such as FO^2, C^2, GF, and GNF?
2. What are the deeper reasons for decidability of conservative extensions in modal and description logics and how far can the limits of decidability be pushed?

To be more precise, we concentrate on *deductive* conservative extensions, that is, $\varphi_1 \wedge \varphi_2$ is a conservative extension of φ_1 if for every sentence ψ formulated in the signature of φ_1, $\varphi_1 \wedge \varphi_2 \models \psi$ implies $\varphi_1 \models \psi$. There is also a *model-theoretic* notion of conservative extension which says that $\varphi_1 \wedge \varphi_2$ is a conservative extension of φ_1 if every model of φ_1 can be extended to a model of φ_2 by interpreting the additional symbols in φ_2. Model-theoretic conservative extensions imply deductive conservative extensions, but the converse fails unless one works with a very expressive logic such as second-order logic [24]. In fact, model-theoretic conservative extensions are undecidable even for some very inexpressive description logics that include neither negation nor disjunction [25]. Deductive conservative extensions, as studied in this paper, are closely related to other important notions in logic, such as uniform interpolation [30, 36, 5]. For example, in logics that enjoy Craig interpolation, a decision procedure for conservative extensions can also be used to decide whether a given sentence φ_2 is a uniform interpolant of a given sentence φ_1 regarding the symbols used in φ_2.

Instead of concentrating only on conservative extensions, we also consider two related reasoning problems that we call Σ-entailment and Σ-inseparability, where Σ denotes a signature. The definitions are as follows: a sentence φ_1 Σ-entails a sentence φ_2 if for every sentence ψ formulated in Σ, $\varphi_2 \models \psi$ implies $\varphi_1 \models \psi$. This can be viewed as a more relaxed notion of conservative extension where it is not required that one sentence actually extends the other as in the conjunction $\varphi_1 \wedge \varphi_2$ used in the definition of conservative extensions. Two sentences φ_1, φ_2 are Σ-inseparable if they Σ-entail each other. We generally prove lower bounds for conservative extensions and upper bounds for Σ-entailment, in this way obtaining the same decidability and complexity results for all three problems.

Our first main result is that conservative extensions are undecidable in FO^2 and (the three-variable fragment of) GF, and in fact in all fragments of FO that contain at least one of the two; note that the latter is not immediate because the separating sentence ψ in the definition of conservative extensions ranges over all sentences from the considered fragment, giving greater separating power when we move to a larger fragment. The proofs are by reductions from the halting problem for two-register machines and a tiling problem, respectively. We note that undecidability of conservative extensions also implies that there is no extension of the logic in question in which consequence is decidable and that has effective uniform interpolation (in the sense that uniform interpolants exist and are computable). We then show as our second main result that, in the two-variable guarded fragment GF^2, Σ-entailment is decidable in 2ExpTime. Regarding the satisfiability problem, GF^2 behaves fairly similarly to modal and description logics. It is thus suprising that deciding Σ-entailment (and conservative extensions) in GF^2 turns out to be much more challenging than in most modal

and description logics. There, the main approach to proving decidability of Σ-entailment is to first establish a suitable model-theoretic characterization based on bisimulations which is then used as a foundation for a decision procedure based on tree automata [27, 7]. In GF^2, an analogous characterization in terms of appropriate guarded bisimulation fails. Instead, one has to demand the existence of *k-bounded* (guarded) bisimulations, *for all* k, and while tree automata can easily handle bisimulations, it is not clear how they can deal with such an infinite family of bounded bisimulations. We solve this problem by a very careful analysis of the situation and by providing another characterization that can be viewed as being 'half way' between a model-theoretic characterization and an automata-encoding of Σ-entailment.

We also observe that a 2ExpTime lower bound from [13] for conservative extensions in description logics can be adapted to GF^2, and consequently our upper bound is tight. It is known that GF^2 enjoys Craig interpolation and thus our results are also relevant to deciding uniform interpolants and to a stronger version of conservative extensions in which the separating sentence ψ can also use 'helper symbols' that occur neither in φ_1 nor in φ_2.

2 Preliminaries

We introduce the fragments of classical first-order logic (FO) that are relevant for this paper. We generally admit equality and disallow function symbols and constants. With FO^2, we denote the *two-variable fragment of FO*, obtained by fixing two variables x and y and disallowing the use of other variables [34, 29]. In FO^2 and fragments thereof, we generally admit only predicates of arity one and two, which is without loss of generality [17]. In the *guarded fragment of FO*, denoted *GF*, quantification is restricted to the pattern

$$\forall \mathbf{y}(\alpha(\mathbf{x},\mathbf{y}) \to \varphi(\mathbf{x},\mathbf{y})) \quad \exists \mathbf{y}(\alpha(\mathbf{x},\mathbf{y}) \wedge \varphi(\mathbf{x},\mathbf{y}))$$

where $\varphi(\mathbf{x},\mathbf{y})$ is a GF formula with free variables among $\mathbf{x},\mathbf{y}$ and $\alpha(\mathbf{x},\mathbf{y})$ is an atomic formula $R\mathbf{xy}$ or an equality $x = y$ that in either case contains all variables in $\mathbf{x},\mathbf{y}$ [1, 16]. The formula α is called the *guard* of the quantifier. The k-variable fragment of GF, defined in the expected way, is denoted GF^k. Apart from the logics introduced so far, in informal contexts we shall also mention several related description logics. Exact definitions are omitted, we refer the reader to [2].

A *signature* Σ is a finite set of predicates. We use $\mathrm{GF}(\Sigma)$ to denote the set of all GF-sentences that use only predicates from Σ (and possibly equality), and likewise for $\mathrm{GF}^2(\Sigma)$ and other fragments. We use $\mathsf{sig}(\varphi)$ to denote the set of predicates that occur in the FO formula φ. Note that we consider equality to be a logical symbol, rather than a predicate, and it is thus never part of a signature. We write $\varphi_1 \models \varphi_2$ if φ_2 is a logical consequence of φ_1. The next definition introduces the central notions studied in this paper.

▶ **Definition 1.** Let F be a fragment of FO, φ_1, φ_2 F-sentences and Σ a signature. Then

1. φ_1 Σ-*entails* φ_2, written $\varphi_1 \models_\Sigma \varphi_2$, if for all $F(\Sigma)$-sentences ψ, $\varphi_2 \models \psi$ implies $\varphi_1 \models \psi$;
2. φ_1 and φ_2 are Σ-*inseparable* if φ_1 Σ-entails φ_2 and vice versa;
3. $\varphi_1 \wedge \varphi_2$ is a *conservative extension* of φ_1 if φ_1 $\mathsf{sig}(\varphi_1)$-entails $\varphi_1 \wedge \varphi_2$.

Note that Σ-entailment could equivalently be defined as follows when F is closed under negation: φ_1 Σ-*entails* φ_2 if for all $F(\Sigma)$-sentences ψ, satisfiability of $\varphi_1 \wedge \psi$ implies satisfiability of $\varphi_2 \wedge \psi$. If φ_1 does not Σ-entail φ_2, there is thus an $F(\Sigma)$-sentence ψ such that $\varphi_1 \wedge \psi$ is satisfiable while $\varphi_2 \wedge \psi$ is not. We refer to such ψ as a *witness sentence* for non-Σ-entailment.

▶ **Example 2.** (1) Σ-entailment is a weakening of logical consequence, that is, $\varphi_1 \models \varphi_2$ implies $\varphi_1 \models_\Sigma \varphi_2$ for any Σ. The converse is true when $\mathsf{sig}(\varphi_2) \subseteq \Sigma$.

(2) Consider the GF^2 sentences $\varphi_1 = \forall x \exists y Rxy$ and $\varphi_2 = \forall x(\exists y(Rxy \wedge Ay) \wedge \exists y(Rxy \wedge \neg Ay))$ and let $\Sigma = \{R\}$. Then $\psi = \forall xy(Rxy \to x = y)$ is a witness for $\varphi_1 \not\models_\Sigma \varphi_2$. If φ_1 is replaced by $\varphi_1' = \forall x \exists y(Rxy \wedge x \neq y)$ we obtain $\varphi_1' \models_\Sigma \varphi_2$ since GF^2 cannot count the number of R-successors.

It is important to note that different fragments F of FO give rise to different notions of Σ-entailment, Σ-inseparability and conservative extensions. For example, if φ_1 and φ_2 belong to GF^2, then they also belong to GF and to FO^2, but it might make a difference whether witness sentences range over all GF^2-sentences, over all GF-sentences, or over all FO^2-sentences. If we want to emphasize the fragment F in which witness sentences are formulated, we speak of $F(\Sigma)$-entailment instead of Σ-entailment and write $\varphi_1 \models_{F(\Sigma)} \varphi_2$, and likewise for $F(\Sigma)$-inseparability and F-conservative extensions.

▶ **Example 3.** Let φ_1', φ_2, and $\Sigma = \{R\}$ be from Example 2 (2). Then φ_1' $\mathrm{GF}^2(\Sigma)$-entails φ_2 but φ_1' does not $\mathrm{FO}(\Sigma)$-entail φ_2; a witness is given by $\forall xy_1y_2((Rxy_1 \wedge Rxy_2) \to y_1 = y_2)$.

Note that conservative extensions and Σ-inseparability reduce in polynomial time to Σ-entailment (with two calls to Σ-entailment required in the case of Σ-inseparability). Moreover, conservative extensions reduce in polynomial time to Σ-inseparability. We thus state our upper bounds in terms of Σ-entailment and lower bounds in terms of conservative extensions.

There is a natural variation of each of the three notions in Definition 1 obtained by allowing to use additional 'helper predicates' in witness sentences. For a fragment F of FO, F-sentences φ_1, φ_2, and a signature Σ, we say that φ_1 *strongly* Σ*-entails* φ_2 if φ_1 Σ'-entails φ_2 for any Σ' with $\Sigma' \cap \mathsf{sig}(\varphi_2) \subseteq \Sigma$. Strong Σ-inseparability and strong conservative extensions are defined accordingly. Strong Σ-entailment implies Σ-entailment, but the converse may fail.

▶ **Example 4.** $\mathrm{GF}(\Sigma)$-entailment does not imply strong $\mathrm{GF}(\Sigma)$-entailment. Let φ_1 state that the binary predicate R is irreflexive and symmetric and let φ_2 be the conjunction of φ_1 and $\forall x(Ax \to \forall y(Rxy \to \neg Ay)) \wedge \forall x(\neg Ax \to \forall y(Rxy \to Ay))$. Thus, an $\{R\}$-structure satisfying φ_1 can be extended to a model of φ_2 if it contains no R-cycles of odd length. Now observe that any satisfiable $\mathrm{GF}(\{R\})$ sentence is satisfiable in a forest $\{R\}$-structure (see Section 4 for a precise definition). Hence, if a $\mathrm{GF}(\{R\})$-sentence is satisfiable in an irreflexive and symmetric structure then it is satisfiable in a structure without odd cycles and so φ_1 $\mathrm{GF}(\{R\})$-entails φ_2. In contrast, for the fresh ternary predicate Q and $\psi = \exists x_1x_2x_3(Qx_1x_2x_3 \wedge Rx_1x_2 \wedge Rx_2x_3 \wedge Rx_3x_1)$ we have $\varphi_2 \models \neg\psi$ but $\varphi_1 \not\models \neg\psi$ and so ψ witnesses that φ_1 does not $\mathrm{GF}(\{R, Q\})$-entail φ_2.

The example above is inspired by proofs that GF does not enjoy Craig interpolation [21, 11]. This is not accidental, as we explain next. Recall that a fragment F of FO *has Craig interpolation* if for all F-sentences ψ_1, ψ_2 with $\psi_1 \models \psi_2$ there exists an F-sentence ψ (called an F*-interpolant for* ψ_1, ψ_2) such that $\psi_1 \models \psi \models \psi_2$ and $\mathsf{sig}(\psi) \subseteq \mathsf{sig}(\psi_1) \cap \mathsf{sig}(\psi_2)$. F *has uniform interpolation* if one can always choose an F-interpolant that does not depend on ψ_2, but only on ψ_1 and $\mathsf{sig}(\psi_1) \cap \mathsf{sig}(\psi_2)$. Thus, given ψ_1, ψ and Σ with $\psi_1 \models \psi$ and $\mathsf{sig}(\psi) \subseteq \Sigma$, then ψ is a *uniform* $F(\Sigma)$*-interpolant of* ψ_1 iff ψ strongly $F(\Sigma)$-entails ψ_1. Both Craig interpolation and uniform interpolation have been investigated extensively, for example for intuitionistic logic [30], modal logics [36, 10, 28], guarded fragments [11], and description logics [27]. The following observation summarizes the connection between (strong) Σ-entailment and interpolation.

▶ **Theorem 5.** *Let F be a fragment of FO that enjoys Craig interpolation. Then $F(\Sigma)$-entailment implies strong $F(\Sigma)$-entailment. In particular, if $\varphi_2 \models \varphi_1$ and $\mathsf{sig}(\varphi_1) \subseteq \Sigma$, then φ_1 is a uniform $F(\Sigma)$-interpolant of φ_2 iff φ_1 $F(\Sigma)$-entails φ_2.*

Proof. Assume φ_1 does not strongly $F(\Sigma)$-entail φ_2. Then there is an F-sentence ψ with $\mathsf{sig}(\psi) \cap \mathsf{sig}(\varphi_2) \subseteq \Sigma$ such that $\varphi_2 \models \psi$ and $\varphi_1 \wedge \neg\psi$ is satisfiable. Let χ be an interpolant for φ_2 and ψ in F. Then $\neg\chi$ witnesses that φ_1 does not $F(\Sigma)$-entail φ_2. ◀

The *uniform interpolant recognition problem for* F is the problem to decide whether a sentence ψ is a uniform $F(\Sigma)$-interpolant of a sentence ψ'. It follows from Theorem 5 that in any fragment F of FO that enjoys Craig interpolation, this problem reduces in polynomial time to Σ-inseparability in F and that, conversely, conservative extension in F reduces in polynomial time to the uniform interpolant recognition problem in F. Neither GF nor FO^2 nor description logics with role inclusions enjoy Craig interpolation [21, 8, 24], but GF^2 does [21]. Thus, our decidability and complexity results for Σ-entailment in GF^2 also apply to strong Σ-entailment and the uniform interpolant recognition problem.

3 Undecidability

We prove that conservative extensions are undecidable in GF^3 and in FO^2, and consequently so are Σ-entailment and Σ-inseparability (as well as strong Σ-entailment and the uniform interpolant recognition problem). These results hold already without equality and in fact apply to all fragments of FO that contain at least one of GF^3 and FO^2 such as the guarded negation fragment [4] and the two-variable fragment with counting quantifiers [19].

We start with the case of GF^3, using a reduction from the halting problem of two-register machines. A (deterministic) *two-register machine (2RM)* is a pair $M = (Q, P)$ with $Q = q_0, \dots, q_\ell$ a set of *states* and $P = I_0, \dots, I_{\ell-1}$ a sequence of *instructions*. By definition, q_0 is the *initial state*, and q_ℓ the *halting state*. For all $i < \ell$,

- either $I_i = +(p, q_j)$ is an *incrementation instruction* with $p \in \{0, 1\}$ a register and q_j the subsequent state;
- or $I_i = -(p, q_j, q_k)$ is a *decrementation instruction* with $p \in \{0, 1\}$ a register, q_j the subsequent state if register p contains 0, and q_k the subsequent state otherwise.

A *configuration* of M is a triple (q, m, n), with q the current state and $m, n \in \mathbb{N}$ the register contents. We write $(q_i, n_1, n_2) \Rightarrow_M (q_j, m_1, m_2)$ if one of the following holds:

- $I_i = +(p, q_j)$, $m_p = n_p + 1$, and $m_{1-p} = n_{1-p}$;
- $I_i = -(p, q_j, q_k)$, $n_p = m_p = 0$, and $m_{1-p} = n_{1-p}$;
- $I_i = -(p, q_k, q_j)$, $n_p > 0$, $m_p = n_p - 1$, and $m_{1-p} = n_{1-p}$.

The *computation* of M on input $(n, m) \in \mathbb{N}^2$ is the unique longest configuration sequence $(p_0, n_0, m_0) \Rightarrow_M (p_1, n_1, m_1) \Rightarrow_M \cdots$ such that $p_0 = q_0$, $n_0 = n$, and $m_0 = m$. The halting problem for 2RMs is to decide, given a 2RM M, whether its computation on input $(0, 0)$ is finite (which implies that its last state is q_ℓ).

We show how to convert a given 2RM M into GF^3-sentences φ_1 and φ_2 such that M halts on input $(0, 0)$ iff $\varphi_1 \wedge \varphi_2$ is not a conservative extension of φ_1. Let $M = (Q, P)$ with $Q = q_0, \dots, q_\ell$ and $P = I_0, \dots, I_{\ell-1}$. We assume w.l.o.g. that $\ell \geq 1$ and that if $I_i = -(p, q_j, q_k)$, then $q_j \neq q_k$. In φ_1, we use the following set Σ of predicates:

- a binary predicate N connecting a configuration to its successor configuration;
- binary predicates R_1 and R_2 that represent the register contents via the length of paths;
- unary predicates $q_0, \dots, q_\ell$ representing the states of M;
- a unary predicate S denoting points where a computation starts.

We define φ_1 to be the conjunction of several GF^2-sentences. First, we say that there is a point where the computation starts:[1]

$$\exists x Sx \wedge \forall x(Sx \to (q_0 x \wedge \neg\exists y\, R_0 xy \wedge \neg\exists y\, R_1 xy))$$

And second, we add that whenever M is not in the final state, there is a next configuration. For $0 \leq i < \ell$:

$$\begin{aligned}
\forall x(q_i x \to \exists y(Nxy \wedge q_j y)) \quad & \text{if } I_i = +(p, q_j) \\
\forall x((q_i x \wedge \neg\exists y R_p xy) \to \exists y(Nxy \wedge q_j y)) \quad & \text{if } I_i = -(p, q_j, q_k) \\
\forall x((q_i x \wedge \exists y R_p xy) \to \exists y(Nxy \wedge q_k y)) \quad & \text{if } I_i = -(p, q_j, q_k)
\end{aligned}$$

The second sentence φ_2 is constructed so as to express that either M does not halt or the representation of the computation of M contains a defect, using the following additional predicates:

- a unary predicate P used to represent that M does not halt;
- binary predicates $D_p^+, D_p^-, D_p^=$ used to describe defects in incrementing, decrementing, and keeping register $p \in \{0, 1\}$;
- ternary predicates $H_1^+, H_2^+, H_1^-, H_2^-, H_1^=, H_2^=$ used as guards for existential quantifiers.

In fact, φ_2 is the disjunction of two sentences. The first sentence says that the computation does not terminate:

$$\exists x\,(Sx \wedge Px) \wedge \forall x\,(Px \to \exists y\,(Nxy \wedge Py))$$

while the second says that registers are not updated properly:

$$\begin{aligned}
\exists x \exists y\,\big(Nxy \wedge \big(& \bigvee_{I_i = +(p,q_j)} (q_i x \wedge q_j y \wedge (D_p^+ xy \vee D_{1-p}^= xy)) \\
& \vee \bigvee_{I_i = -(p,q_j,q_k)} (q_i x \wedge q_k y \wedge (D_p^- xy \vee D_{1-p}^= xy)) \\
& \vee \bigvee_{I_i = -(p,q_j,q_k)} (q_i x \wedge q_j y \wedge (D_p^= xy \vee D_{1-p}^= xy))\big)\big) \\
\wedge \forall x \forall y\,(D_p^+ xy \to & (\neg\exists z\, R_p yz \vee (\neg\exists z\, R_p xz \wedge \exists z\,(R_p yz \wedge \exists x R_p zx)) \\
& \vee \exists z(H_1^+ xyz \wedge R_p xz \wedge \exists x(H_2^+ xzy \wedge R_p yx \wedge D_p^+ zx)).
\end{aligned}$$

In this second sentence, additional conjuncts that implement the desired behaviour of $D_p^=$ and D_p^- are also needed; they are constructed analogously to the last three lines above (but using guards H_j^- and $H_j^=$), details are omitted. The following is proved in the appendix of the full version of this paper.

▶ **Lemma 6.**

1. *If M halts, then $\varphi_1 \wedge \varphi_2$ is not a GF^2-conservative extension of φ_1.*
2. *If there exists a Σ-structure that satisfies φ_1 and cannot be extended to a model of φ_2 (by interpreting the predicates in $\mathsf{sig}(\varphi_2) \setminus \mathsf{sig}(\varphi_1)$), then M halts.*

In the proof of Point 1, the sentence that witnesses non-conservativity describes a halting computation of M, up to global $\mathrm{GF}^2(\Sigma)$-bisimulations. This can be done using only two variables. The following result is an immediate consequence of Lemma 6.

[1] The formulas that are not syntactically guarded can easily be rewritten into such formulas.

▶ **Theorem 7.** *In any fragment of FO that extends the three-variable guarded fragment GF^3, the following problems are undecidable: conservative extensions, Σ-inseparability, Σ-entailment, and strong Σ-entailment.*

Since Point 1 of Lemma 6 ensures GF^2-witnesses, Theorem 7 can actually be strengthened to say that GF^2-conservative extensions of GF^3-sentences are undecidable.

Our result for FO^2 is proved by a reduction of a tiling problem that asks for the tiling of a rectangle (of any size) such that the borders are tiled with certain distinguished tiles. Because of space limitations, we defer details to the appendix of the full version and state only the obtained result.

▶ **Theorem 8.** *In any fragment of FO that extends FO^2, the following problems are undecidable: conservative extensions, Σ-inseparability, Σ-entailment, and strong Σ-entailment.*

It is interesting to note that the proof of Theorem 8 also shows that FO^2-conservative extensions of $\mathcal{ALC}$-TBoxes are undecidable while it follows from our results below that GF^2-conservative extensions of $\mathcal{ALC}$-TBoxes are decidable.

4 Characterizations

The undecidability results established in the previous section show that neither the restriction to two variables nor guardedness alone are sufficient for decidability of conservative extensions and related problems. In the remainder of the paper, we show that adopting both restrictions simultaneously results in decidability of Σ-entailment (and thus also of conservative extensions and of inseparability). We proceed by first establishing a suitable model-theoretic characterization and then use it as the foundation for a decision procedure based on tree automata. We in fact establish two versions of the characterization, the second one building on the first one.

We start with some preliminaries. An *atomic 1-type for* Σ is a maximal satisfiable set τ of atomic $\mathrm{GF}^2(\Sigma)$-formulas and their negations that use the variable x, only. We use $\mathrm{at}^\Sigma_{\mathfrak{A}}(a)$ to denote the atomic 1-type for Σ realized by the element a in the structure $\mathfrak{A}$. An *atomic 2-type for* Σ is a maximal satisfiable set τ of atomic $\mathrm{GF}^2(\Sigma)$-formulas and their negations that use the variables x and y, only, and contains $\neg(x = y)$. We say that τ is *guarded* if it contains an atom of the form Rxy or Ryx, R a predicate symbol. We use $\mathrm{at}^\Sigma_{\mathfrak{A}}(a, b)$ to denote the atomic 2-type for Σ realized by the elements a, b in the structure $\mathfrak{A}$. A relation $\sim \subseteq A \times B$ is a *$GF^2(\Sigma)$-bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$* if the following conditions hold whenever $a \sim b$:

1. $\mathrm{at}^\Sigma_{\mathfrak{A}}(a) = \mathrm{at}^\Sigma_{\mathfrak{B}}(b)$;
2. for every $a' \neq a$ such that $\mathrm{at}^\Sigma_{\mathfrak{A}}(a, a')$ is guarded, there is a $b' \neq b$ such that $\mathrm{at}^\Sigma_{\mathfrak{A}}(a, a') = \mathrm{at}^\Sigma_{\mathfrak{B}}(b, b')$ and $a' \sim b'$ (forth condition);
3. for every $b' \neq b$ such that $\mathrm{at}^\Sigma_{\mathfrak{B}}(b, b')$ is guarded, there is an $a' \neq a$ such that $\mathrm{at}^\Sigma_{\mathfrak{A}}(a, a') = \mathrm{at}^\Sigma_{\mathfrak{B}}(b, b')$ and $a' \sim b'$ (back condition).

We write $(\mathfrak{A}, a) \sim_\Sigma (\mathfrak{B}, b)$ and say that $(\mathfrak{A}, a)$ and $(\mathfrak{B}, b)$ are *$GF^2(\Sigma)$-bisimilar* if there is a $\mathrm{GF}^2(\Sigma)$-bisimulation $\sim$ between $\mathfrak{A}$ and $\mathfrak{B}$ with $a \sim b$. If the domain and range of $\sim$ coincide with A and B, respectively, then $\sim$ is called a *global $GF^2(\Sigma)$-bisimulation*.

We next introduce a bounded version of bisimulations. For $k \geq 0$, we write $(\mathfrak{A}, a) \sim^k_\Sigma (\mathfrak{B}, b)$ and say that $(\mathfrak{A}, a)$ and $(\mathfrak{B}, b)$ are *k-$GF^2(\Sigma)$-bisimilar* if there is a $\sim \subseteq A \times B$ such that the first condition for bisimulations holds and the back and forth conditions can be iterated up to k times starting from a and b; a formal definition is in the appendix of the full version. It is straightforward to show the following link between k-GF^2-bisimilarity and GF^2-sentences of guarded quantifier depth k (defined in the obvious way).

▶ **Lemma 9.** *Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures, Σ a signature, and $k \geq 0$. Then the following conditions are equivalent:*

1. *for all $a \in A$ there exists $b \in B$ with $(\mathfrak{A}, a) \sim^k_\Sigma (\mathfrak{B}, b)$ and vice versa;*
2. *$\mathfrak{A}$ and $\mathfrak{B}$ satisfy the same $GF^2(\Sigma)$-sentences of guarded quantifier depth at most k.*

The corresponding lemma for $\mathrm{GF}^2(\Sigma)$-sentences of unbounded guarded quantifier depth and $\mathrm{GF}^2(\Sigma)$-bisimulations holds if $\mathfrak{A}$ and $\mathfrak{B}$ satisfy certain saturation conditions (for example, if $\mathfrak{A}$ and $\mathfrak{B}$ are ω-saturated). It can then be proved that an FO-sentence φ is equivalent to a GF^2 sentence iff its models are preserved under global $\mathrm{GF}^2(\mathsf{sig}(\varphi))$-bisimulations [18, 14]. In modal and description logic, global Σ-bisimulations can often be used to characterize Σ-entailment in the following natural way [27]: φ_1 Σ-entails φ_2 iff every for every (tree) model $\mathfrak{A}$ of φ_1, there exists a (tree) model $\mathfrak{B}$ of φ_2 that is globally Σ-bisimilar to $\mathfrak{A}$. Such a characterization enables decision procedures based on tree automata, but does not hold for GF^2.

▶ **Example 10.** Let $\varphi_1 = \forall x \exists y Rxy$ and let $\varphi_2 = \varphi_1 \wedge \exists x Bx \wedge \forall x(Bx \to \exists y(Ryx \wedge By))$. Let $\mathfrak{A}$ be the model of φ_1 that consists of an infinite R-path with an initial element. Then there is no model of φ_2 that is globally $\mathrm{GF}^2(\{R\})$-bisimilar to $\mathfrak{A}$ since any such model has to contain an infinite R-path with no initial element. Yet, φ_2 is a conservative extension of φ_1 which can be proved using Theorem 11 below.

We give our first characterization theorem that uses unbounded bisimulations in one direction and bounded bisimulations in the other.

▶ **Theorem 11.** *Let φ_1, φ_2 be GF_2-sentences and Σ a signature. Then $\varphi_1 \models_\Sigma \varphi_2$ iff for every model $\mathfrak{A}$ of φ_1 of finite outdegree, there is a model $\mathfrak{B}$ of φ_2 such that*

1. *for every $a \in A$ there is a $b \in B$ such that $(\mathfrak{A}, a) \sim_\Sigma (\mathfrak{B}, b)$*
2. *for every $b \in B$ and every $k \geq 0$, there is an $a \in A$ such that $(\mathfrak{A}, a) \sim^k_\Sigma (\mathfrak{B}, b)$.*

The direction ($\Leftarrow$) follows from Lemma 9 and ($\Rightarrow$) can be proved using compactness and ω-saturated structures. Because of the use of k-bounded bisimulations (for unbounded k), it is not clear how to use Theorem 11 to find a decision procedure based on tree automata. In the following, we formulate a more 'operational' but also more technical characterization that no longer mentions bounded bisimulations. It additionally refers to forest models $\mathfrak{A}$ of φ_1 (of finite outdegree) instead of unrestricted models, but we remark that Theorem 11 also remains true under this modification.

A structure $\mathfrak{A}$ is a *forest* if its Gaifman graph is a forest. Thus, a forest admits cycles of length one and two, but not of any higher length. A (Σ-)*tree* in a forest structure $\mathfrak{A}$ is a maximal (Σ)-connected substructure of $\mathfrak{A}$. When working with forest structures $\mathfrak{A}$, we will typically view them as directed forests rather than as undirected ones. This can be done by choosing a root for each tree in the Gaifman graph of $\mathfrak{A}$, thus giving rise to notions such as successor, descendant, etc. Which node is chosen as the root will always be irrelevant. Note that the direction of binary relations does not need to reflect the successor relation. When speaking of a *path* in a forest structure $\mathfrak{A}$, we mean a path in the directed sense; when speaking of a *subtree*, we mean a tree that is obtained by choosing a root a and restricting the structure to a and its descendants. We say that $\mathfrak{A}$ is *regular* if it has only finitely many subtrees, up to isomorphism.

To see how we can get rid of bounded bisimulations, reconsider Theorem 11. The characterization is still correct if we pull out the quantification over k in Point 2 so that the theorem reads '...iff for every model $\mathfrak{A}$ of φ_1 of finite outdegree and every $k \geq 0$, there is...'. In fact, this modified version of Theorem 11 is even closer to the definition of Σ-entailment. It

also suggests that we add a marking $A_\perp \subseteq A$ of elements in $\mathfrak{A}$, representing 'break-off points' for bisimulations, and then replace k-bisimulations with bisimulations that stop whenever they have encountered the *second* marked element on the same path—in this way, the distance between marked elements (roughly) corresponds to the bound k in k-bisimulations. However, we would need a marking $A_\perp$, for any $k \geq 0$, such that there are infinitely many markers on any infinite path and the distance between any two markers in a tree is at least k. It is easy to see that such a marking may not exist, for example when $k = 3$ and $\mathfrak{A}$ is the infinite full binary tree. We solve this problem as follows. First, we only demand that the distance between any two markers *on the same path* is at least k. And second, we use the markers only when following bisimulations upwards in a tree while downwards, we use unbounded bisimulations. This does not compromise correctness of the characterization.

We next introduce a version of bisimulations that implement the ideas just explained. Let $\mathfrak{A}$ and $\mathfrak{B}$ be forest models, Σ a signature, and $A_\perp \subseteq A$. Two relations $\sim_\Sigma^{A_\perp,0}, \sim_\Sigma^{A_\perp,1} \subseteq A \times B$ form an $A_\perp$*-delimited* $GF^2(\Sigma)$*-bisimulation* between $\mathfrak{A}$ and $\mathfrak{B}$ if the following conditions are satisfied:

1. if $(\mathfrak{A}, a) \sim_\Sigma^{A_\perp,0} (\mathfrak{B}, b)$, then $\mathrm{at}_\mathfrak{A}^\Sigma(a) = \mathrm{at}_\mathfrak{B}^\Sigma(b)$ and
   a. for every $a' \neq a$ with $\mathrm{at}_\mathfrak{A}^\Sigma(a, a')$ guarded, there is a $b' \neq b$ such that $(\mathfrak{A}, a') \sim_\Sigma^{A_\perp,i} (\mathfrak{B}, b')$ where $i = 1$ if a' is the predecessor of a and $a' \in A_\perp$, and $i = 0$ otherwise;
   b. for every $b' \neq b$ with $\mathrm{at}_\mathfrak{B}^\Sigma(b, b')$ guarded, there is an $a' \neq a$ such that $(\mathfrak{A}, a') \sim_\Sigma^{A_\perp,i} (\mathfrak{B}, b')$ where $i = 1$ if a' is the predecessor of a and $a' \in A_\perp$, and $i = 0$ otherwise;
2. if $(\mathfrak{A}, a) \sim_\Sigma^{A_\perp,1} (\mathfrak{B}, b)$ and the predecessor of a in $\mathfrak{A}$ is not in $A_\perp$, then $\mathrm{at}_\mathfrak{A}^\Sigma(a) = \mathrm{at}_\mathfrak{B}^\Sigma(b)$ and
   a. for every $a' \neq a$ with $\mathrm{at}_\mathfrak{A}^\Sigma(a, a')$ guarded, there is a $b' \neq b$ such that $(\mathfrak{A}, a') \sim_\Sigma^{A_\perp,i} (\mathfrak{B}, b')$ where $i = 0$ if a is the predecessor of a' and $a \in A_\perp$, and $i = 1$ otherwise;
   b. for every $b' \neq b$ with $\mathrm{at}_\mathfrak{B}^\Sigma(b, b')$ guarded, there is an $a' \neq a$ such that $(\mathfrak{A}, a') \sim_\Sigma^{A_\perp,i} (\mathfrak{B}, b')$ where $i = 0$ if a is the predecessor of a' and $a \in A_\perp$, and $i = 1$ otherwise.

Then $(\mathfrak{A}, a)$ and $(\mathfrak{B}, b)$ are $A_\perp$*-delimited* $GF^2(\Sigma)$*-bisimilar*, in symbols $(\mathfrak{A}, a) \sim_\Sigma^{A_\perp} (\mathfrak{B}, b)$, if there exists an $A_\perp$-delimited $\mathrm{GF}^2(\Sigma)$-bisimulation $\sim_\Sigma^{A_\perp,0}, \sim_\Sigma^{A_\perp,1}$ between $\mathfrak{A}$ and $\mathfrak{B}$ such that $(\mathfrak{A}, a) \sim_\Sigma^{A_\perp,0} (\mathfrak{B}, b)$.

Let φ be a GF^2-sentence. We use $\mathsf{cl}(\varphi)$ to denote the set of all subformulas of φ closed under single negation and renaming of free variables (using only the available variables x and y). A 1*-type for* φ is a subset $t \subseteq \mathsf{cl}(\varphi)$ that contains only formulas of the form $\psi(x)$ and such that $\varphi \wedge \exists x \bigwedge t(x)$ is satisfiable. For a model $\mathfrak{A}$ of φ and $a \in A$, we use $\mathsf{tp}_\mathfrak{A}(a)$ to denote the 1-type $\{\psi(x) \in \mathsf{cl}(\varphi) \mid \mathfrak{A} \models \psi(a)\}$, assuming that φ is understood from the context. We say that the 1-type t is *realized* in $\mathfrak{A}$ if there is an $a \in A$ with $\mathsf{tp}_\mathfrak{A}(a) = t$. We are now ready to formulate our final characterizations.

▶ **Theorem 12.** *Let φ_1, φ_2 be GF^2-sentences and Σ a signature. Then $\varphi_1 \models_\Sigma \varphi_2$ iff for every regular forest model $\mathfrak{A}$ of φ_1 that has finite outdegree and for every set $A_\perp \subseteq A$ with $A_\perp \cap \rho$ infinite for any infinite Σ-path ρ in $\mathfrak{A}$, there is a model $\mathfrak{B}$ of φ_2 such that*

1. *for every $a \in A$, there is a $b \in B$ such that $(\mathfrak{A}, a) \sim_\Sigma (\mathfrak{B}, b)$;*
2. *for every 1-type t for φ_2 that is realized in $\mathfrak{B}$, there are $a \in A$ and $b \in B$ such that $\mathsf{tp}_\mathfrak{B}(b) = t$ and $(\mathfrak{A}, a) \sim_\Sigma^{A_\perp} (\mathfrak{B}, b)$.*

Regularity and finite outdegree are used in the proof of Theorem 12 given in the appendix of the full version, but it follows from the automata constructions below that the theorem is still correct when these qualifications are dropped.

5 Decidability and Complexity

We show that Σ-entailment in GF^2 is decidable and 2ExpTime-complete, and thus so are conservative extensions and Σ-inseparability. The upper bound is based on Theorem 12 and uses alternating parity automata on infinite trees. Since Theorem 12 does not provide us with an obvious upper bound on the outdegree of the involved tree models, we use alternating tree automata which can deal with trees of any finite outdegree, similar to the ones introduced by Wilke [37], but with the capability to move both downwards and upwards in the tree.

A *tree* is a non-empty (and potentially infinite) set of words $T \subseteq (\mathbb{N} \setminus 0)^*$ closed under prefixes. We generally assume that trees are finitely branching, that is, for every $w \in T$, the set $\{i \mid w \cdot i \in T\}$ is finite. For any $w \in (\mathbb{N} \setminus 0)^*$, as a convention we set $w \cdot 0 := w$. If $w = n_0 n_1 \cdots n_k$, we additionally set $w \cdot -1 := n_0 \cdots n_{k-1}$. For an alphabet Θ, a Θ-*labeled tree* is a pair (T, L) with T a tree and $L : T \to \Theta$ a node labeling function.

A *two-way alternating tree automata (2ATA)* is a tuple $\mathcal{A} = (Q, \Theta, q_0, \delta, \Omega)$ where Q is a finite set of *states*, Θ is the *input alphabet*, $q_0 \in Q$ is the *initial state*, δ is a *transition function* as specified below, and $\Omega : Q \to \mathbb{N}$ is a *priority function*, which assigns a priority to each state. The transition function maps a state q and some input letter $\theta \in \Theta$ to a *transition condition* $\delta(q, \theta)$ which is a positive Boolean formula over the truth constants true and false and transitions of the form q, $\langle - \rangle q$, $[-]q$, $\Diamond q$, $\Box q$ where $q \in Q$. The automaton runs on Θ-labeled trees. Informally, the transition q expresses that a copy of the automaton is sent to the current node in state q, $\langle - \rangle q$ means that a copy is sent in state q to the predecessor node, which is then required to exist, $[-]q$ means the same except that the predecessor node is not required to exist, $\Diamond q$ means that a copy is sent in state q to some successor, and $\Box q$ that a copy is sent in state q to all successors. The semantics is defined in terms of runs in the usual way, we refer to the appendix of the full version for details. We use $L(\mathcal{A})$ to denote the set of all Θ-labeled trees accepted by $\mathcal{A}$. It is standard to verify that 2ATAs are closed under complementation and intersection. We show in the appendix that the emptiness problem for 2ATAs can be solved in time exponential in the number of states.

We aim to show that given two GF^2-sentences φ_1 and φ_2 and a signature Σ, one can construct a 2ATA $\mathcal{A}$ such that $L(\mathcal{A}) = \emptyset$ iff $\varphi_1 \models_{\mathrm{GF}^2(\Sigma)} \varphi_2$. The number of states of the 2ATA $\mathcal{A}$ is polynomial in the size of φ_1 and exponential in the size of φ_2, which yields the desired 2ExpTime upper bounds.

Let φ_1, φ_2, and Σ be given. Since the logics we are concerned with have Craig interpolation, we can assume w.l.o.g. that $\Sigma \subseteq \mathsf{sig}(\varphi_1)$. With Θ, we denote the set of all pairs (τ, M) where τ is an atomic 2-type for $\mathsf{sig}(\varphi_1)$ and $M \in \{0, 1\}$. For $p = (\tau, M) \in \Theta$, we use p^1 to denote τ and p^2 to denote M. A Θ-labeled tree (T, L) represents a forest structure $\mathfrak{A}_{(T,L)}$ with universe $A_{(T,L)} = T$ and where $w \in A^{\mathfrak{A}_{(T,L)}}$ if $A(y) \in L(w)$ and $(w, w') \in R^{\mathfrak{A}_{(T,L)}}$ if one of the following conditions is satisfied: (1) $w = w'$ and $Ryy \in L(w)^1$; (2) w' is a successor of w and $Rxy \in L(w')^1$; (3) w is a successor of w' and $Ryx \in L(w)^1$. Thus, the atoms in a node label that involve only the variable y describe the current node, the atoms that involve both variables x and y describe the connection between the predecessor and the current node, and the atoms that involve only the variable x are ignored. The M-components of node labels are used to represent a set of markers $A_\perp = \{w \in A_{(T,L)} \mid L(w)^2 = 1\}$. It is easy to see that, conversely, for every tree structure $\mathfrak{A}$ over Σ, there is a Θ-labeled tree (T, L) such that $\mathfrak{A}_{(T,L)} = \mathfrak{A}$.

To obtain the desired 2ATA $\mathcal{A}$, we construct three 2ATAs $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ and then define $\mathcal{A}$ so that it accepts $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)} \cap L(\mathcal{A}_3)$. The 2ATA $\mathcal{A}_3$ only makes sure that the set $A_\perp \subseteq A_{(T,L)}$ is such that for any infinite Σ-path ρ, $A_\perp \cap \rho$ is infinite (as required by

Theorem 12), we omit details. We construct $\mathcal{A}_1$ so that it accepts a Θ-labeled tree (T, L) iff $\mathfrak{A}_{(T,L)}$ is a model of φ_1. The details of the construction, which is fairly standard, can be found in the appendix. The number of states of $\mathcal{A}_1$ is polynomial in the size of φ_1 and independent of φ_2. The most interesting automaton is $\mathcal{A}_2$.

▶ **Lemma 13.** *There is a 2ATA $\mathcal{A}_2$ that accepts a Θ-labeled tree (T, L) iff there is a model $\mathfrak{B}$ of φ_2 s.t. Conditions 1 and 2 from Theorem 12 are satisfied when $\mathfrak{A}$ is replaced with $\mathfrak{A}_{(T,L)}$.*

The general idea of the construction of $\mathcal{A}_2$ is to check the existence of the desired model $\mathfrak{B}$ of φ_2 by verifying that there is a set of 1-types for φ_2 from which $\mathfrak{B}$ can be assembled, represented via the states that occur in a successful run. Before we can give details, we introduce some preliminaries.

A 0-*type* s *for* φ_2 is a maximal set of sentences $\psi() \in \mathsf{cl}(\varphi_2)$ such that $\varphi_2 \wedge s$ is satisfiable. A 2-*type* λ *for* φ_2 is a maximal set of formulas $\psi(x, y) \in \mathsf{cl}(\varphi_2)$ that contains $\neg(x = y)$ and such that $\varphi_2 \wedge \exists xy\, \lambda(x, y)$ is satisfiable. If a 2-type λ contains the atom Rxy or Ryx for at least one binary predicate R, then it is *guarded*. If additionally $R \in \Sigma$, then it is Σ-*guarded*. Note that each 1-type contains a (unique) 0-type and each 2-type contains two (unique) 1-types. Formally, we use λ_x to denote the 1-type obtained by restricting the 2-type λ to the formulas that do not use the variable y, and likewise for λ_y and the variable x. We use TP_n to denote the set of n-types for φ_2, $n \in \{0, 1, 2\}$. For $t \in \mathsf{TP}_1$ and a $\lambda \in \mathsf{TP}_2$, we say that λ is *compatible with* t and write $t \approx \lambda$ if the sentence $\varphi_2 \wedge \exists xy(t(x) \wedge \lambda(x, y))$ is satisfiable; for $t \in \mathsf{TP}_1$ and $T \subseteq \mathsf{TP}_2$ a set of guarded 2-types, we say that T is a *neighborhood for* t and write $t \approx T$ if the sentence

$$\varphi_2 \wedge \exists x\big(t(x) \wedge \bigwedge_{\lambda \in T} \exists y\, \lambda(x, y) \wedge \forall y \bigvee_{R \in \mathsf{sig}(\varphi_2)} ((Rxy \vee Ryx) \to \bigvee_{\lambda \in T} \lambda(x, y))\big)$$

is satisfiable. Note that each of the mentioned sentences is formulated in GF^2 and at most single exponential in size (in the size of φ_1 and φ_2), thus satisfiability can be decided in 2ExpTime.

To build the automaton $\mathcal{A}_2$ from Lemma 13, set $\mathcal{A}_2 = (Q_2, \Theta, q_0, \delta_2, \Omega_2)$ where Q_2 is

$$\{q_0, q_\bot\} \cup \mathsf{TP}_0 \cup \{t, t^?, t_\uparrow, t_\downarrow, t_\&, t^i, t^i_\uparrow, t^i_\downarrow \mid t \in \mathsf{TP}_1,\ i \in \{0, 1\}\} \cup$$
$$\{\lambda, \lambda_\uparrow, \lambda^i, \lambda^i_\uparrow \mid \lambda \in \mathsf{TP}_2,\ i \in \{0, 1\}\},$$

Ω_2 assigns two to all states except for those of the form $t^?$, to which it assigns one.

The automaton begins by choosing the 0-type s realized in the forest model $\mathfrak{B}$ of φ_2 whose existence it aims to verify. For every $\exists x\varphi(x) \in s$, it then chooses a 1-type t in which $\varphi(x)$ is realized in $\mathfrak{B}$ and sends off a copy of itself to find a node where t is realized. To satisfy Condition 1 of Theorem 12, at each node it further chooses a 1-type that is compatible with s, to be realized at that node. This is implemented by the following transitions:

$$\begin{aligned}
\delta_2(q_0, \sigma) &= \bigvee_{s \in \mathsf{TP}_0} \big(s \wedge \bigwedge_{\exists x\, \varphi(x) \in s} \bigvee_{\substack{t \in \mathsf{TP}_1 \mid \\ s \cup \{\varphi(x)\} \subseteq t}} t^?\big) \\
\delta_2(s, \sigma) &= \Box s \wedge \bigvee_{t \in \mathsf{TP}_1, s \subseteq t} t \\
\delta_2(t^?, \sigma) &= \langle -1 \rangle t^? \vee \Diamond t^? \vee t^0
\end{aligned}$$

where s ranges over TP_0. When a state of the form t is assigned to a node w, this is an obligation to prove that there is a $\mathrm{GF}^2(\Sigma)$-bisimulation between the element w in $\mathfrak{A}_{(T,L)}$ and

an element b of type t in $\mathfrak{B}$. A state of the form t^0 represents the obligation to verify that there is an $A_\bot$-delimited $\mathrm{GF}^2(\Sigma)$-bisimulation between w and an element of type t in $\mathfrak{B}$. We first verify that the former obligations are satisfied. This requires to follow all successors of w and to guess types of successors of b to be mapped there, satisfying the back condition of bisimulations. We also need to guess successors of b in $\mathfrak{B}$ (represented as a neighborhood for t) to satisfy the existential demands of t and then select successors of a to which they are mapped, satisfying the "back" condition of bisimulations. Whenever we decide to realize a 1-type t in $\mathfrak{B}$ that does not participate in the bisimulation currently being verified, we also send another copy of the automaton in state $t^?$ to guess an $a \in A_{(T,L)}$ that we can use to satisfy Condition 2 from Theorem 12:

$$\delta_2(t,(\tau,M)) \;=\; t_\uparrow \wedge \Box t_\downarrow \wedge \bigvee_{T \mid t \approx T} \bigwedge_{\lambda \in T} (\Diamond\lambda \vee \lambda_\uparrow) \quad \text{if } \tau_y =_\Sigma t$$

$$\begin{array}{rcll}
\delta_2(t,(\tau,M)) &=& \mathsf{false} & \text{if } \tau_y \neq_\Sigma t \\
\delta_2(t_\downarrow,(\tau,M)) &=& \mathsf{true} & \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_\downarrow,(\tau,M)) &=& \displaystyle\bigvee_{\lambda \mid t \approx \lambda \wedge \tau =_\Sigma \lambda} \lambda_y & \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(t_\uparrow,(\tau,M)) &=& \mathsf{true} & \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_\uparrow,(\tau,M)) &=& \displaystyle\bigvee_{\lambda \mid t \approx \lambda \wedge \tau =_\Sigma \lambda^-} [-1]\lambda_y & \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(\lambda,(\tau,M)) &=& \lambda_y & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_\Sigma \lambda \\
\delta_2(\lambda,(\tau,M)) &=& \mathsf{false} & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_\Sigma \lambda \\
\delta_2(\lambda,(\tau,M)) &=& \lambda^?_y & \text{if } \lambda \text{ is not } \Sigma\text{-guarded} \\
\delta_2(\lambda_\uparrow,(\tau,M)) &=& \langle -1\rangle \lambda_y & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_\Sigma \lambda^- \\
\delta_2(\lambda_\uparrow,(\tau,M)) &=& \mathsf{false} & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_\Sigma \lambda^- \\
\delta_2(\lambda_\uparrow,(\tau,M)) &=& \lambda^?_y & \text{if } \lambda \text{ is not } \Sigma\text{-guarded}
\end{array}$$

where $\tau_y =_\Sigma t$ means that the atoms in τ that mention only y are identical to the Σ-relational atoms in t (up to renaming x to y), $\tau =_\Sigma \lambda$ means that the restriction of λ to Σ-atoms is exactly τ, and λ^- is obtained from λ by swapping x and y. We need further transitions to satisfy the obligations represented by states of the form t^0, which involves checking $A_\bot$-delimited bisimulations. Details are given in the appendix where also the correctness of the construction is proved.

▶ **Theorem 14.** *In GF^2, Σ-entailment and conservative extensions can be decided in time $2^{2^{p(|\varphi_2| \cdot \log |\varphi_1|)}}$, for some polynomial p. Moreover, Σ-inseparability is in* 2ExpTime.

Note that the time bound for conservative extensions given in Theorem 14 is double exponential only in the size of φ_2 (that is, the extension). In ontology engineering applications, φ_2 will often be small compared with φ_1.

A matching lower bound is proved using a reduction of the word problem of exponentially space-bounded alternating Turing machines, see the appendix for details. The construction is inspired by the proof from [13] that conservative extensions in the description logic $\mathcal{ALC}$ are 2ExpTime-hard, but the lower bound does not transfer directly since we are interested here in witness sentences that are formulated in GF^2 rather than in $\mathcal{ALC}$.

▶ **Theorem 15.** *In any fragment of FO that contains GF^2, the problems Σ-entailment, Σ-inseparability, conservative extensions, and strong Σ-entailment are* 2ExpTime*-hard.*

6 Conclusion

We have shown that conservative extensions are undecidable in (extensions of) GF and FO^2, and that they are decidable and 2ExpTime-complete in GF^2. It thus appears that decidability of conservative extensions is linked even more closely to the tree model property than decidability of the satisfiability problem: apart from cycles of length at most two, GF^2 enjoys a 'true' tree model property while GF only enjoys a bounded treewidth model property and FO^2 has a rather complex regular model property that is typically not even made explicit. As future work, it would be interesting to investigate whether conservative extensions remain decidable when guarded counting quantifiers, transitive relations, equivalence relations, or fixed points are added, see e.g. [32, 22, 20]. It would also be interesting to investigate a finite model version of conservative extensions.

References

1 Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philosophical Logic*, 27(3):217–274, 1998.

2 Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003. (2nd edition, 2007).

3 Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. *Logical Methods in Computer Science*, 10(2), 2014.

4 Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, 2015.

5 Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Interpolation with decidable fixpoint logics. In *Proc. of LICS*, pages 378–389. IEEE Computer Society, 2015.

6 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

7 Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Inseparability and conservative extensions of description logic ontologies: A survey. In *Proc. of Reasoning Web*, volume 9885 of *LNCS*, pages 27–89. Springer, 2016.

8 Stephen D. Comer. Classes without the amalgamation property. *Pacific Journal of Mathematics*, 28:309–318, 1969.

9 Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research (JAIR)*, 31:273–318, 2008.

10 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the μ-calculus: Interpolation, Lyndon and ł oś-Tarski. *J. Symb. Log.*, 65(1):310–332, 2000.

11 Giovanna D'Agostino and Giacomo Lenzi. Bisimulation quantifiers and uniform interpolation for guarded first order logic. *Theor. Comput. Sci.*, 563:75–85, 2015.

12 Răzvan Diaconescu, Joseph A. Goguen, and Petros Stefaneas. Logical support for modularisation. In Gerard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130, 1993.

13 Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I damage my ontology? A case for conservative extensions in description logic. In *Proc. of KR*, pages 187–197. AAAI Press, 2006.

14 Valentin Goranko and Martin Otto. Model theory of modal logic. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, pages 249–330. Elsevier, 2006.

15 Michael J. C. Gordon and Thomas F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

16 Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.

17 Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

18 Erich Grädel and Martin Otto. The freedoms of (guarded) bisimulation. In Alexandru Baltag and Sonja Smets, editors, *Johan van Benthem on Logic and Information Dynamics*, pages 3–31. Springer, 2014.

19 Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *Proc. of LICS*, pages 306–317. IEEE Computer Society, 1997.

20 Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *Proc. of LICS*, pages 45–54. IEEE Computer Society, 1999.

21 Eva Hoogland and Maarten Marx. Interpolation and definability in guarded fragments. *Studia Logica*, 70(3):373–409, 2002.

22 Emanuel Kieronski. On the complexity of the two-variable guarded fragment with transitive guards. *Inf. Comput.*, 204(11):1663–1703, 2006.

23 Emanuel Kieronski, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. *SIAM J. Comput.*, 43(3):1012–1063, 2014.

24 Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Formal properties of modularisation. In Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors, *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *LNCS*, pages 25–66. Springer, 2009.

25 Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203:66–103, 2013.

26 Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In *IJCAI*, pages 453–458, 2007.

27 Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proc. of IJCAI*, pages 989–995. IJCAI/AAAI, 2011.

28 Johannes Marti, Fatemeh Seifan, and Yde Venema. Uniform interpolation for coalgebraic fixpoint logic. In *CALCO*, volume 35 of *LIPIcs*, pages 238–252. Schloss Dagstuhl, 2015.

29 Michael Mortimer. On languages with two variables. *Math. Log. Q.*, 21(1):135–140, 1975.

30 Andrew M. Pitts. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. of Symbolic Logic*, 57, 1992.

31 Stephen Pollard. *Philosophical Introduction to Set Theory*. University of Notre Dame Press, 1990.

32 Ian Pratt-Hartmann. Complexity of the guarded two-variable fragment with counting quantifiers. *J. Log. Comput.*, 17(1):133–155, 2007.

33 Ian Pratt-Hartmann. Data-complexity of the two-variable fragment with counting quantifiers. *Inf. Comput.*, 207(8):867–888, 2009.

34 Dana Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:1962, 1962.

35 Stephen G. Simpson. *Subsystems of Second Order Arithmetic*. Cambridge University Press, 2009.

36 Albert Visser. Uniform interpolation and layered bisimulation. In *Gödel'96 (Brno, 1996)*, volume 6 of *Lecture Notes in Logic*, pages 139–164. Springer, 1996.

37 Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8(2), 2001.

Models and Termination of Proof Reduction in the $\lambda\Pi$-Calculus Modulo Theory

Gilles Dowek

Inria and École normale supérieure de Paris-Saclay, Cachan Cedex, France
gilles.dowek@ens-paris-saclay.fr

Abstract

We define a notion of model for the $\lambda\Pi$-calculus modulo theory and prove a soundness theorem. We then define a notion of super-consistency and prove that proof reduction terminates in the $\lambda\Pi$-calculus modulo any super-consistent theory. We prove this way the termination of proof reduction in several theories including Simple type theory and the Calculus of constructions.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases model, proof reduction, Simple type theory, Calculus of constructions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.109

1 Introduction

1.1 Models and termination

In Predicate logic, a model is defined by a domain $\mathcal{M}$, a set $\mathcal{B}$ of truth values, and an interpretation function, parametrized by a valuation ϕ, mapping each term t to an element $[\![t]\!]_\phi$ of $\mathcal{M}$, and each proposition A to an element $[\![A]\!]_\phi$ of $\mathcal{B}$.

Predicate logic can be extended to Deduction modulo theory [11, 12], where a congruence on propositions defining a computational equality, also known as definitional equality in Constructive type theory [17], is added. Proofs of a proposition A are then considered to also be proofs of any proposition congruent to A. In Deduction modulo theory, like in Predicate logic, a model is defined by a domain $\mathcal{M}$, a set $\mathcal{B}$ of truth values, and an interpretation function.

Usually, the set $\mathcal{B}$ is the two-element set $\{0, 1\}$, but the notion of model can be extended to a notion of many-valued model, where $\mathcal{B}$ is an arbitrary Boolean algebra, a Heyting algebra, a pre-Boolean algebra [5], or a pre-Heyting algebra [9]. Boolean algebras permit to introduce intermediate truth values for propositions that are neither provable nor disprovable, Heyting algebras to construct models of constructive logic, and pre-Boolean and pre-Heyting algebras, where the order relation $\leq$ is replaced by a pre-order relation, to distinguish a notion of weak equivalence: $[\![A]\!]_\phi \leq [\![B]\!]_\phi$ and $[\![B]\!]_\phi \leq [\![A]\!]_\phi$, for all ϕ, from a notion of strong equivalence: $[\![A]\!]_\phi = [\![B]\!]_\phi$, for all ϕ. In Deduction modulo theory, the first corresponds to the provability of $A \Leftrightarrow B$ and the second to the congruence.

In a model valued in a Boolean algebra, a Heyting algebra, a pre-Boolean algebra, or a pre-Heyting algebra, a proposition A is said to be valid when it is weakly equivalent to the proposition $\top$, that is when, for all ϕ, $[\![A]\!]_\phi \geq \tilde{\top}$, and this condition can be rephrased as $[\![A]\!]_\phi = \tilde{\top}$ in Boolean and Heyting algebras. A congruence $\equiv$ defined on propositions is said to be valid when, for all A and B such that $A \equiv B$, A and B are strongly equivalent, that is, for all ϕ, $[\![A]\!]_\phi = [\![B]\!]_\phi$. Note that the relation $\leq$ is used in the definition of the validity of a proposition, but not in the definition of the validity of a congruence.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 109; pp. 109:1–109:14


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

Proof reduction terminates in Deduction modulo a theory defined by a set of axioms $\mathcal{T}$ and a congruence $\equiv$, when this theory has a model valued in the pre-Heyting algebra of reducibility candidates [12]. As a consequence, proof reduction terminates if the theory is super-consistent, that is if, for all pre-Heyting algebras $\mathcal{B}$, it has a model valued in $\mathcal{B}$ [9]. This theorem permits to completely separate the semantic and the syntactic aspects that are often mixed in the usual proofs of termination of proof reduction. The semantic aspect is in the proof of super-consistency of the considered theory and the syntactic in the universal proof that super-consistency implies termination of proof reduction.

For the termination of proof reduction, the congruence matters, but the axioms do not. Thus, the pre-order relation $\leq$ does not matter in the algebra of reducibility candidates and it is possible to define it as the trivial pre-order relation such that $C \leq C'$, for all C and C'. Such a pre-Heyting algebra is said to be trivial. As the pre-order is trivial, all the conditions defining pre-Heyting algebras, such as $a \tilde{\wedge} b \leq a$, $a \tilde{\wedge} b \leq b$... are always satisfied in a trivial pre-Heyting algebra, and a trivial pre-Heyting algebra is just a set equipped with arbitrary operations $\tilde{\wedge}$, $\tilde{\Rightarrow}$... Thus, in order to prove that proof reduction terminates in Deduction modulo a theory defined by a set of axioms $\mathcal{T}$ and a congruence $\equiv$, it is sufficient to prove that for all trivial pre-Heyting algebras $\mathcal{B}$, the theory has a model valued in $\mathcal{B}$.

1.2 The λΠ-calculus modulo theory

In Predicate logic and in Deduction modulo theory, terms, propositions, and proofs belong to three distinct languages. But, it is more thrifty to consider a single language, such as the λΠ-calculus modulo theory [8], which is implemented in the Dedukti system [1], or Martin-Löf's Logical Framework [21], and express terms, propositions, and proofs, in this language. For instance, in Predicate logic, 0 is a term, $P(0) \Rightarrow P(0)$ is a proposition and $\lambda\alpha : P(0)\ \alpha$ is a proof of this proposition. In the λΠ-calculus modulo theory, all these expressions are terms of the calculus. Only their types differ: 0 has type nat, $P(0) \Rightarrow P(0)$ has type $Type$ and $\lambda\alpha : P(0)\ \alpha$ has type $P(0) \Rightarrow P(0)$.

Like the λΠ-calculus, the λΠ-calculus modulo theory is a λ-calculus with dependent types, but, like in Deduction modulo theory, its conversion rule is extended to an arbitrary congruence, typically defined with a confluent and terminating rewrite system. This idea of extending the conversion rule beyond β-reduction is already present in Martin-Löf type theory. It is used, in various ways, in different systems [20, 6, 13, 3].

1.3 From pre-Heyting algebras to Π-algebras

The first goal of this paper is to extend the notion of pre-Heyting algebra to a notion of Π-algebra, adapted to the λΠ-calculus modulo theory.

In Predicate logic and in Deduction modulo theory, the propositions are built from atomic propositions with the connectors and quantifiers $\top$, $\bot$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, and $\exists$. Accordingly, the operations of a pre-Heyting algebra are $\tilde{\top}$, $\tilde{\bot}$, $\tilde{\wedge}$, $\tilde{\vee}$, $\tilde{\Rightarrow}$, $\tilde{\forall}$, and $\tilde{\exists}$. In the λΠ-calculus and in the λΠ-calculus modulo theory, the only connector is Π. Thus, a Π-algebra mainly has an operation $\tilde{\Pi}$. As expected, its properties are a mixture of the properties of the implication and of the universal quantifier of the pre-Heyting algebras.

1.4 Layered models

The second goal of this paper is to extend the usual notion of model to the λΠ-calculus modulo theory.

Extending the notion of model to many-sorted predicate logic requires to consider not just one domain $\mathcal{M}$, but a family of domains $\mathcal{M}_s$ indexed by the sorts. For instance, in a model of Simple type theory, the family of domains is indexed by simple types. In the $\lambda\Pi$-calculus modulo theory, the sorts also are just terms of the calculus. Thus, we shall define a model of the $\lambda\Pi$-calculus modulo theory by a family of domains $(\mathcal{M}_t)_t$ indexed by the terms of the calculus and a function $[\![.]\!]$ mapping each term t of type A and valuation ϕ to an element $[\![t]\!]_\phi$ of $\mathcal{M}_A$.

The functions $\mathcal{M}$ and $[\![.]\!]$ are similar, in the sense that both their domains is the set of terms of the calculus. The goal of the model construction is to define the function $[\![.]\!]$ and the function $\mathcal{M}$ can be seen as a tool helping to define this function. For instance, if f is a constant of type $A \to A$, where A is a term of type $Type$, and we have the rule $f(x) \longrightarrow x$, we want to define the interpretation $[\![f]\!]$ as the identity function over some set, but to state which, we must first define the function $\mathcal{M}$ that maps the term A to a set $\mathcal{M}_A$, and then define $[\![f]\!]$ as the identity function over the set $\mathcal{M}_A$.

In Predicate logic and in Deduction modulo theory, terms may be typed with sorts, but the sorts themselves have no type. In the $\lambda\Pi$-calculus modulo theory, in contrast, terms have types that have types... This explains that, in some cases, constructing the function $\mathcal{M}$ itself requires to define first another function $\mathcal{N}$, that is used as a tool helping to define this function. This can be iterated to a several layer model, where the function $[\![.]\!]$ is defined with the help of a function $\mathcal{M}$, that is defined with the help of a function $\mathcal{N}$, that is defined with the help... The number of layers depends on the model. Such layered constructions are common in proofs of termination of proof reduction [14, 18, 4], for instance for Pure Type Systems where sorts are stacked: $Type_0 : Type_1 : Type_2 : Type_3$.

Note that, in this definition of the notion of model, when a term t has type A, we do not require $[\![t]\!]_\phi$ to be an element of $[\![A]\!]_\phi$, but of $\mathcal{M}_A$. This is consistent with the notion of model of many-sorted predicate logic, where we require $[\![t]\!]_\phi$ to be an element of $\mathcal{M}_s$ and where $[\![s]\!]_\phi$ is often not even defined.

Valuations must be handled with care in such layered models. In a three layer model, for instance, the definition of $\mathcal{N}_t$ is absolute, the definition of $\mathcal{M}_t$ is relative to a valuation ψ, mapping each variable of type A to an element of $\mathcal{N}_A$, and the definition of $[\![t]\!]$ is relative to a valuation ψ and to a valuation ϕ mapping each variable of type A to an element of $\mathcal{M}_{A,\psi}$.

1.5 Super-consistency and proof reduction

The third goal of this paper is to use this notion of Π-algebra to define a notion of super-consistency and to prove that proof reduction, that is β-reduction, terminates in the $\lambda\Pi$-calculus modulo any super-consistent theory.

We prove this way the termination of proof reduction in several theories expressed in the $\lambda\Pi$-calculus modulo theory, including Simple type theory [11] and the Calculus of constructions [8]. Together with confluence, this termination of proof reduction is a property required to define these theories in the system Dedukti [1].

In Section 2, we recall the definition of the $\lambda\Pi$-calculus modulo theory and give three examples of theories expressed in this framework. In Section 3, we introduce the notion of Π-algebra and that of model for the $\lambda\Pi$-calculus modulo theory and we prove a soundness theorem. In Section 4, we define the notion of super-consistency and prove that the three theories introduced in Section 2 are super-consistent. In Section 5, we prove that proof reduction terminates in the $\lambda\Pi$-calculus modulo any super-consistent theory.

$$\frac{}{[\,]\ \text{well-formed}}\ \textbf{Empty}$$

$$\frac{\Gamma \vdash A : Type}{\Gamma, x : A\ \text{well-formed}}\ x \notin \Gamma\ \textbf{Declaration (for object variables)}$$

$$\frac{\Gamma \vdash A : Kind}{\Gamma, x : A\ \text{well-formed}}\ x \notin \Gamma\ \textbf{Declaration (for type family variables)}$$

$$\frac{\Gamma\ \text{well-formed}}{\Gamma \vdash Type : Kind}\ \textbf{Sort}$$

$$\frac{\Gamma\ \text{well-formed}}{\Gamma \vdash x : A}\ x : A \in \Gamma\ \textbf{Variable}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}\ \textbf{Product (for types)}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Kind}{\Gamma \vdash \Pi x : A\ B : Kind}\ \textbf{Product (for kinds)}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction (for objects)}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Kind \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\ \textbf{Abstraction (for type families)}$$

$$\frac{\Gamma \vdash t : \Pi x : A\ B \quad \Gamma \vdash u : A}{\Gamma \vdash (t\ u) : (u/x)B}\ \textbf{Application}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : Type \quad \Gamma \vdash t : A}{\Gamma \vdash t : B}\ A \equiv_\beta B\ \textbf{Conversion (for types)}$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma \vdash B : Kind \quad \Gamma \vdash t : A}{\Gamma \vdash t : B}\ A \equiv_\beta B\ \textbf{Conversion (for kinds)}$$

Figure 1 The λΠ-calculus.

2 The λΠ-calculus modulo theory

2.1 The λΠ-calculus

The syntax of the λΠ-calculus is

$$t = x \mid Type \mid Kind \mid \Pi x : t\ t \mid \lambda x : t\ t \mid t\ t$$

and the typing rules are given in Figure 1.

As usual, we write $A \to B$ for $\Pi x : A\ B$ when x does not occur in B. The α-equivalence relation is defined as usual and terms are identified modulo α-equivalence. The relation β – one step β-reduction at the root – is defined as usual. If r is a relation on terms, we write $\longrightarrow_r^1$ for the congruence closure of r, $\longrightarrow_r^+$ for the transitive closure of $\longrightarrow_r^1$, $\longrightarrow_r^*$ for its reflexive-transitive closure, and $\equiv_r$ for its reflexive-symmetric-transitive closure.

If Σ, Γ, and Δ are contexts, a substitution θ, binding the variables of Γ, is said to *have type* $\Gamma \rightsquigarrow \Delta$ *in* Σ if for all $x : A$ in Γ, we have $\Sigma, \Delta \vdash \theta x : \theta A$. In this case, if $\Sigma, \Gamma \vdash t : B$, then $\Sigma, \Delta \vdash \theta t : \theta B$.

Types are preserved by β-reduction. The β-reduction relation is confluent and strongly terminating. And each term has a unique type modulo β-equivalence [16].

A term t, well-typed in some context Γ, is a *kind* if its type in this context is $Kind$. For instance, $Type$ and $nat \to Type$ are kinds. It is a *type family* if its type is a kind. In particular, it is a *type* if its type is $Type$. For instance, nat, $array$, and $(array\ 0)$ are type families, among which nat and $(array\ 0)$ are types. It is an *object* if its type is a type. For instance, 0 and [0] are objects.

$$
\begin{array}{rcl}
\iota & : & Type \\
o & : & Type \\
\Rightarrow & : & o \to o \to o \\
\forall_A & : & (A \to o) \to o \\
\varepsilon & : & o \to Type \\
\\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall_A\ x)) & \longrightarrow & \Pi z : A\ (\varepsilon\ (x\ z))
\end{array}
$$

with a finite number of quantifiers $\forall_A$

Figure 2 Simple type theory.

2.2 The λΠ-calculus modulo theory

▶ **Definition 1** (Rewrite rule). A *rewrite rule* is a triple $l \longrightarrow^{\Gamma} r$ where Γ is a context and l and r are β-normal terms. Such a rule is *well-typed* in the context Σ if, in the $\lambda\Pi$-calculus, the context Σ, Γ is well-formed and there exists a term A such that the terms l and r both have type A in this context.

If Σ is a context, $l \longrightarrow^{\Gamma} r$ is a rewrite rule well-typed in Σ and θ is a substitution of type $\Gamma \rightsquigarrow \Delta$ in Σ, then the terms θl and θr both have type θA in the context Σ, Δ. The relation $\mathcal{R}$ – one step $\mathcal{R}$-reduction at the root – is defined by: $t\ \mathcal{R}\ u$ is there exists a rewrite rule $l \longrightarrow^{\Gamma} r$ and a substitution θ such that $t = \theta l$ and $u = \theta r$. The relation $\beta\mathcal{R}$ – one step $\beta\mathcal{R}$-reduction at the root – is the union of β and $\mathcal{R}$.

▶ **Definition 2** (Theory). A *theory* is a pair formed with a context Σ, well-formed in the $\lambda\Pi$-calculus, and a set of rewrite rules $\mathcal{R}$, well-typed in Σ in the $\lambda\Pi$-calculus.

The variables declared in Σ are called *constants*. They replace the sorts, the function symbols, the predicate symbols, and the axioms of a theory in Predicate logic.

▶ **Definition 3** (The λΠ-calculus modulo theory). The $\lambda\Pi$-*calculus modulo* $\Sigma, \mathcal{R}$ is the extension of the $\lambda\Pi$-calculus obtained modifying the **Declaration** rules to replace the condition $x \notin \Gamma$ with $x \notin \Sigma, \Gamma$, the **Variable** rules to replace the condition $x : A \in \Gamma$ by $x : A \in \Sigma, \Gamma$, and the **Conversion** rules to replace the condition $A \equiv_{\beta} B$ with $A \equiv_{\beta\mathcal{R}} B$.

In this paper, we assume that the relation $\longrightarrow^{1}_{\beta\mathcal{R}}$ is confluent and has the subject reduction property. Confluence and subject reduction are indeed needed to build models and prove termination of proof reduction. This is consistent with the methodology proposed in [2]: first prove confluence and subject reduction, then termination.

2.3 Examples of theories

Simple type theory can be expressed in Deduction modulo theory [10]. The main idea in this presentation is to distinguish terms of type o from propositions. If t is a term of type o, the corresponding proposition is written $\varepsilon(t)$. The term t is a *propositional content* or a *code* of the proposition $\varepsilon(t)$. This way, it is not possible to quantify over propositions, but it is possible to quantify over codes of propositions: there is no proposition $\forall X\ (X \Rightarrow X)$, but there is a proposition $\forall x\ (\varepsilon(x) \Rightarrow \varepsilon(x))$, respecting the syntax of Predicate logic, where the predicate symbol ε is applied to the variable x to form a proposition. In this presentation,

$$
\begin{array}{rcl}
type & : & Type \\
\iota & : & type \\
o & : & type \\
arrow & : & type \to type \to type \\
\eta & : & type \to Type \\
\Rightarrow & : & (\eta\ o) \to (\eta\ o) \to (\eta\ o) \\
\forall & : & \Pi a : type\ (((\eta\ a) \to (\eta\ o)) \to (\eta\ o)) \\
\varepsilon & : & (\eta\ o) \to Type \\
 & & \\
(\eta\ (arrow\ x\ y)) & \longrightarrow & (\eta\ x) \to (\eta\ y) \\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z))
\end{array}
$$

Figure 3 Simple type theory with a parametric quantifier.

each simple type is a sort and, for each simple type A, there is a quantifier $\forall_A$. Thus, the language contains an infinite number of sorts and an infinite number of constants.

This presentation can be adapted to the λΠ-calculus modulo theory. To avoid declaring an infinite number of constants for simple types, we can just declare two constants ι and o of type $Type$ and use the product of the λΠ-calculus modulo theory to represent the simple types $\iota \to \iota$, $\iota \to \iota \to \iota$, $\iota \to o$... We should declare an infinite number of quantifiers $\forall_A$, indexed by simple types, but this can be avoided as, in each specific proof, only a finite number of such quantifiers occur. This leads to the theory presented in Figure 2.

Another possibility is to add the type A as an extra argument of the quantifier $\forall$. To do so, we need to introduce a type $type$ for codes of simple types, two constants ι and o, of type $type$, and not $Type$, a constant $arrow$ of type $type \to type \to type$, and a decoding function η of type $type \to Type$. This way, the quantifier $\forall$ can be given the type $\Pi a : type\ (((\eta\ a) \to (\eta\ o)) \to (\eta\ o))$. This leads to the theory presented in Figure 3.

The Calculus of constructions [7] can also be expressed in the λΠ-calculus modulo theory [8] as the theory presented in Figure 4.

3 Algebras and Models

3.1 Π-algebras

The notion of Π-algebra is an adaptation of that of pre-Heyting algebra to the λΠ-calculus.

▶ **Definition 4** (Π-algebra)**.** A Π-algebra is formed with

- a set $\mathcal{B}$,
- a pre-order relation $\leq$ on $\mathcal{B}$,
- an element $\tilde{\top}$ of $\mathcal{B}$,
- a function $\tilde{\wedge}$ from $\mathcal{B} \times \mathcal{B}$ to $\mathcal{B}$,
- a subset $\mathcal{A}$ of $\mathcal{P}^+(\mathcal{B})$, the set of non-empty subsets of $\mathcal{B}$,
- a function $\tilde{\Pi}$ from $\mathcal{B} \times \mathcal{A}$ to $\mathcal{B}$,

such that

- $\tilde{\top}$ is a maximal element for $\leq$, that is for all a in $\mathcal{B}$, $a \leq \tilde{\top}$,
- $a\ \tilde{\wedge}\ b$ is a greatest lower bound of $\{a, b\}$ for $\leq$, that is $a\ \tilde{\wedge}\ b \leq a$, $a\ \tilde{\wedge}\ b \leq b$, and for all c, if $c \leq a$ and $c \leq b$, then $c \leq a\ \tilde{\wedge}\ b$,
- $a \leq \tilde{\Pi}(b, S)$ if and only if for all c in S, $a\ \tilde{\wedge}\ b \leq c$.

A Π-algebra is *full* if $\mathcal{A} = \mathcal{P}^+(\mathcal{B})$, that is if $\tilde{\Pi}$ is total on $\mathcal{B} \times \mathcal{P}^+(\mathcal{B})$.

$$
\begin{array}{rcl}
type & : & Type \\
o & : & type \\
\eta & : & type \to Type \\
\varepsilon & : & (\eta\ o) \to Type \\
\dot{\Pi}_{KK} & : & \Pi x : type\ (((\eta\ x) \to type) \to type) \\
\dot{\Pi}_{TT} & : & \Pi x : (\eta\ o)\ (((\varepsilon\ x) \to (\eta\ o)) \to (\eta\ o)) \\
\dot{\Pi}_{KT} & : & \Pi x : type\ (((\eta\ x) \to (\eta\ o)) \to (\eta\ o)) \\
\dot{\Pi}_{TK} & : & \Pi x : (\eta\ o)\ (((\varepsilon\ x) \to type) \to type)
\end{array}
$$

$$
\begin{array}{rcl}
(\eta\ (\dot{\Pi}_{KK}\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\eta\ (y\ z)) \\
(\varepsilon\ (\dot{\Pi}_{TT}\ x\ y)) & \longrightarrow & \Pi z : (\varepsilon\ x)\ (\varepsilon\ (y\ z)) \\
(\varepsilon\ (\dot{\Pi}_{KT}\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z)) \\
(\eta\ (\dot{\Pi}_{TK}\ x\ y)) & \longrightarrow & \Pi z : (\varepsilon\ x)\ (\eta\ (y\ z))
\end{array}
$$

Figure 4 The Calculus of constructions.

Note that is the relation $\leq$ is a pre-order, and not necessarily an order, greatest lower bounds are not necessarily unique, when they exist.

Note also that, from the operation $\tilde{\Pi}$, we can define an exponentiation operation $b \tilde{\to} c = \tilde{\Pi}(b, \{c\})$ that verifies the usual properties of exponentiation: $a \leq b \tilde{\to} c$ if and only if $a\ \tilde{\wedge}\ b \leq c$. When the set S has a greatest lower bound $\tilde{\bigwedge} S$, the operation mapping b and S to $b \tilde{\to} \tilde{\bigwedge} S$ verifies the same properties as $\tilde{\Pi}$: $a \leq b \tilde{\to} \tilde{\bigwedge} S$ if and only if $a\ \tilde{\wedge}\ b \leq \tilde{\bigwedge} S$ if and only if for all c in S, $a\ \tilde{\wedge}\ b \leq c$. But this decomposition is possible only when all sets of $\mathcal{A}$ have greatest lower bounds.

▶ **Example 5.** The algebra $\langle \{0,1\}, 1, \tilde{\wedge}, \mathcal{P}^+(\{0,1\}), \tilde{\Pi} \rangle$, where $\tilde{\wedge}$ and $\tilde{\Pi}$ are defined by the tables below, is a Π-algebra. Note that, dropping the middle column of the table of $\tilde{\Pi}$, we get the table of implication and, dropping the first line, that of the universal quantifier.

$\tilde{\wedge}$	0	1
0	0	0
1	0	1

$\tilde{\Pi}$	$\{0\}$	$\{0,1\}$	$\{1\}$
0	1	1	1
1	0	0	1

3.2 Models valued in a Π-algebra $\mathcal{B}$

▶ **Definition 6** (Model). A *model* is a family of interpretation functions $\mathcal{D}^1, ..., \mathcal{D}^n$ such that for all i, $\mathcal{D}^i$ is a function mapping each term t of type B in some context Γ, function ϕ_1 mapping each variable $x : A$ of Γ to an element of $\mathcal{D}^1_A$, ..., and function ϕ_{i-1} mapping each variable $x : A$ of Γ to an element of $\mathcal{D}^{i-1}_{A,\phi_1,...,\phi_{n-2}}$, to some $\mathcal{D}^i_{t,\phi_1,...,\phi_{i-1}}$ in $\mathcal{D}^{i-1}_{B,\phi_1,...,\phi_{i-2}}$, and for all t, u, ϕ_1, ..., ϕ_{n-1}

$$\mathcal{D}^n_{(u/x)t,\phi_1,...,\phi_{n-1}} = \mathcal{D}^n_{t,(\phi_1,x=\mathcal{D}^1_u),...,(\phi_{n-1},x=\mathcal{D}^{n-1}_{u,\phi_1,...,\phi_{n-2}})}$$

For the last function $\mathcal{D}^n$, we write $[\![t]\!]_{\phi_1,...,\phi_{n-1}}$ instead of $\mathcal{D}^n_{t,\phi_1,...,\phi_{n-1}}$.

In the examples presented in this paper, we use the cases $n = 2$ and $n = 3$ only. The general definition then specializes as follows.

▶ **Example 7.** When $n = 2$, a model is given by two functions $\mathcal{M}$ and $[\![.]\!]$ such that
- $\mathcal{M}$ is a function mapping each term t of type B in Γ to some $\mathcal{M}_t$,
- $[\![.]\!]$ is a function mapping each term t of type B in Γ and function ϕ mapping each variable $x : A$ of Γ to an element of $\mathcal{M}_A$, to some $[\![t]\!]_\phi$ in $\mathcal{M}_B$, such that for all t, u and ϕ

$$[\![(u/x)t]\!]_\phi = [\![t]\!]_{\phi, x=[\![u]\!]_\phi}.$$

This generalizes of the usual definition of *model* for many-sorted predicate logic.

Note that if f is a constant of type $A \to A \to A$, we can define the function $\hat{f}$ mapping a and b in $\mathcal{M}_A$ to $[\![(f\ x\ y)]\!]_{x=a,y=b}$. Using the property $[\![(u/x)t]\!]_\phi = [\![t]\!]_{\phi,x=[\![u]\!]_\phi}$, we then get $[\![(f\ t\ u)]\!]_\phi = \hat{f}([\![t]\!]_\phi, [\![u]\!]_\phi)$, which is the usual definition of an interpretation.

Note also that the first interpretation function $\mathcal{M}$ does not depend on any valuation, so it must be very rudimentary. For instance in Definition 15 below, for all objects and most types, we have $\mathcal{M}_t = \{e\}$. Only the types o, $o \to o$... are interpreted in a non trivial way. Nevertheless, it is sufficient to support the definition of the function $[\![.]\!]$.

▶ **Example 8.** When $n = 3$, a model is given by three functions $\mathcal{N}$, $\mathcal{M}$, and $[\![.]\!]$ such that
- $\mathcal{N}$ is a function mapping each term t of type B in Γ to some $\mathcal{N}_t$,
- $\mathcal{M}$ is a function mapping each term t of type B in Γ and function ψ mapping each variable $x : A$ of Γ to an element of $\mathcal{N}_A$, to some $\mathcal{M}_{t,\psi}$ in $\mathcal{N}_B$,
- $[\![.]\!]$ is a function mapping each term t of type B in Γ, function ψ mapping each variable $x : A$ of Γ to an element of $\mathcal{N}_A$, and function ϕ mapping each variable $x : A$ of Γ to an element of $\mathcal{M}_{A,\psi}$, to some $[\![t]\!]_{\psi,\phi}$ in $\mathcal{M}_{B,\psi}$, such that for all t, u, ψ, and ϕ

$$[\![(u/x)t]\!]_{\psi,\phi} = [\![t]\!]_{(\psi, x=\mathcal{M}_{u,\psi}),(\phi, x=[\![u]\!]_{\psi,\phi})}.$$

▶ **Definition 9** (Model valued in a Π-algebra $\mathcal{B}$)**.** Let $\mathcal{B} = \langle \mathcal{B}, \tilde{\top}, \tilde{\wedge}, \mathcal{A}, \tilde{\Pi} \rangle$ be a Π-algebra. A model is *valued in* $\mathcal{B}$ if
- $\mathcal{D}^{n-1}_{Kind,\phi_1,...,\phi_{n-2}} = \mathcal{D}^{n-1}_{Type,\phi_1,...,\phi_{n-2}} = \mathcal{B}$,
- $[\![Kind]\!]_{\phi_1,...,\phi_{n-1}} = [\![Type]\!]_{\phi_1,...,\phi_{n-1}} = \tilde{\top}$,
- $[\![\Pi x : C\ D]\!]_{\phi_1,...,\phi_{n-1}} = \tilde{\Pi}([\![C]\!]_{\phi_1,...,\phi_{n-1}}, \{[\![D]\!]_{(\phi_1,x=c_1),...,(\phi_{n-1},x=c_{n-1})} \mid c_1 \in \mathcal{D}^1_C, ..., c_{n-1} \in \mathcal{D}^{n-1}_{C,\phi_1,...,\phi_{n-2}}\})$.

We often write $\overline{\phi}$ for a sequence $\phi_1, ..., \phi_n$ and, if $\overline{c} = c_1, ..., c_n$, we write $\overline{\phi}, x = \overline{c}$ for the sequence $(\phi_1, x = c_1), ..., (\phi_n, x = c_n)$.

▶ **Definition 10** (Validity)**.** A model $\mathcal{M}$ valued in some Π-algebra $\mathcal{B}$ is *model* of a theory $\Sigma, \mathcal{R}$, or the theory is *valid* in the model, if
- for all constants $c : A$ in Σ, we have $[\![A]\!] \geq \tilde{\top}$,
- and for all A and B well-typed in a context Γ, such that $A \equiv_{\beta\mathcal{R}} B$, we have for all i, for all $\overline{\phi}$, $\mathcal{D}^i_{A,\overline{\phi}} = \mathcal{D}^i_{B,\overline{\phi}}$.

▶ **Theorem 11** (Soundness)**.** *Let $\mathcal{M}$ be a model, valued in some Π-algebra $\mathcal{B}$, of a theory $\Sigma, \mathcal{R}$. Then, for all judgments $x_1 : A_1, ..., x_p : A_p \vdash t : B$ derivable in $\Sigma, \mathcal{R}$, and for all $\overline{\phi}$, we have*

$$[\![A_1]\!]_{\overline{\phi}} \tilde{\wedge} ... \tilde{\wedge} [\![A_p]\!]_{\overline{\phi}} \leq [\![B]\!]_{\overline{\phi}}.$$

▶ **Corollary 12** (Consistency)**.**
- *Let $\mathcal{M}$ be a model, valued in some Π-algebra $\mathcal{B}$, of a theory $\Sigma, \mathcal{R}$. Then, for all judgments $\vdash t : B$ derivable in $\Sigma, \mathcal{R}$, we have $[\![B]\!]_{\overline{\phi}} \geq \tilde{\top}$.*

- *Let $\mathcal{M}$ be a model, valued in the two-element* Π*-algebra of Example 5, of a theory* $\Sigma, \mathcal{R}$*. Then, for all judgments* $\vdash t : B$*, derivable in this theory, we have* $[\![B]\!]_{\overline{\phi}} = 1$*.*
- *Let* $\Sigma, \mathcal{R}$ *be a theory that has a model, valued in the two-element* Π*-algebra of Example 5. Then, there is no term* t *such that the judgment* $P : Type \vdash t : P$ *is derivable in* Σ, Γ*.*

4 Super-consistency

4.1 Super-consistency

We now want to define a notion of notion of *super-consistency*: a theory is super-consistent if for every Π-algebra, there exists a model of this theory valued in this algebra.

Unfortunately, this constraint is sometimes too strong, as it does not allow to define interpretations as fixed points, for instance if we have a rule $P \longrightarrow ((P \Rightarrow Q) \Rightarrow Q)$, we want to define the interpretation of P as the fixed point of the function mapping b to $(b \mathrel{\tilde{\Rightarrow}} a) \mathrel{\tilde{\Rightarrow}} a$, where a is the interpretation of Q, but this function does not have a fixed point in all Π-algebras. Thus, we weaken this constraint, requiring the existence of model for *complete* Π-algebras only. Defining this notion of completeness requires to introduce an order relation $\sqsubseteq$, that need not be related to the pre-order $\leq$.

▶ **Definition 13** (Ordered, complete Π-algebra)**.** A Π-algebra is *ordered* if it is equipped with an order relation $\sqsubseteq$ such that the operation $\tilde{\Pi}$ is left anti-monotonic and right monotonic with respect to $\sqsubseteq$, that is

- if $a \sqsubseteq b$, then for all S, $\tilde{\Pi}(b, S) \sqsubseteq \tilde{\Pi}(a, S)$,
- if $S \sqsubseteq T$, then for all a, $\tilde{\Pi}(a, S) \sqsubseteq \tilde{\Pi}(a, T)$,

where the relation $\sqsubseteq$ is extended to sets of elements of $\mathcal{B}$ in a trivial way: $S \sqsubseteq T$ if for all a in S, there exists a b in T such that $a \sqsubseteq b$.

It is *complete* if every subset of $\mathcal{B}$ has a least upper bound for the relation $\sqsubseteq$.

▶ **Definition 14** (Super-consistency)**.** A theory $\Sigma, \mathcal{R}$, is *super-consistent* if, for every full, ordered and complete Π-algebra $\mathcal{B}$, there exists a model $\mathcal{M}$, valued in $\mathcal{B}$, of $\Sigma, \mathcal{R}$.

We now prove that the three theories presented in Section 2.3 are super-consistent.

4.2 Simple type theory

Let $\mathcal{B} = \langle \mathcal{B}, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{B}), \tilde{\Pi} \rangle$ be a full Π-algebra. We construct a model of Simple type theory, valued in $\mathcal{B}$, in two steps. The first is the construction of the interpretation function $\mathcal{M}$ and the second the construction of the interpretation function $[\![.]\!]$. The key idea in this construction is to take $\mathcal{M}_o = \mathcal{B}$, to interpret ε as the identity over $\mathcal{B}$, and $\Rightarrow$ like $\rightarrow$ in order to validate the rewrite rule

$$\varepsilon\ (\Rightarrow\ \ x\ y) \longrightarrow (\varepsilon\ x) \rightarrow (\varepsilon\ y)\,.$$

Let S and T be two sets, we write $\mathcal{F}(S, T)$ for the set of functions from S to T. Let $\{e\}$ be an arbitrary one-element set such that e is not in $\mathcal{B}$.

▶ **Definition 15** (A model of Simple type theory)**.**

- $\mathcal{M}_{Kind} = \mathcal{M}_{Type} = \mathcal{B}$,
- $\mathcal{M}_{\Pi x:C\ D} = \mathcal{F}(\mathcal{M}_C, \mathcal{M}_D)$, except if $\mathcal{M}_D = \{e\}$, in which case $\mathcal{M}_{\Pi x:C\ D} = \{e\}$,
- $\mathcal{M}_\iota = \mathcal{M}_\Rightarrow = \mathcal{M}_{\forall_A} = \mathcal{M}_\varepsilon = \{e\}$,
- $\mathcal{M}_o = \mathcal{B}$,

- $\mathcal{M}_x = \{e\}$,
- $\mathcal{M}_{\lambda x:C\ t} = \mathcal{M}_t$,
- $\mathcal{M}_{(t\ u)} = \mathcal{M}_t$.

- $[\![Kind]\!]_\phi = [\![Type]\!]_\phi = \tilde{\top}$,
- $[\![\Pi x : C\ D]\!]_\phi = \tilde{\Pi}([\![C]\!]_\phi, \{[\![D]\!]_{\phi,x=c} \mid c \in \mathcal{M}_C\})$,
- $[\![\iota]\!]_\phi = \tilde{\top}$,
- $[\![o]\!]_\phi = \tilde{\top}$,
- $[\![\Rightarrow]\!]_\phi$ is the function mapping a and b in $\mathcal{B}$ to $\tilde{\Pi}(a, \{b\})$,
- $[\![\forall_C]\!]_\phi$ is the function mapping f in $\mathcal{F}(\mathcal{M}_C, \mathcal{B})$ to $\tilde{\Pi}([\![C]\!]_\phi, \{(f\ c) \mid c \in \mathcal{M}_C\})$,
- $[\![\varepsilon]\!]_\phi$ is the identity on $\mathcal{B}$,
- $[\![x]\!]_\phi = \phi x$,
- $[\![\lambda x : C\ t]\!]_\phi$ is the function mapping c in $\mathcal{M}_C$ to $[\![t]\!]_{\phi,x=c}$, except if for all c in $\mathcal{M}_C$, $[\![t]\!]_{\phi,x=c} = e$ in which case $[\![\lambda x : C\ t]\!]_\phi = e$,
- $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi\ [\![u]\!]_\phi$, except if $[\![t]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = e$.

We prove that if $\Gamma \vdash t : B$, and ϕ is a function mapping variables $x : A$ of Γ to elements of $\mathcal{M}_A$, then $[\![t]\!]_\phi \in \mathcal{M}_B$. That for all t, u and ϕ, $[\![(u/x)t]\!]_\phi = [\![t]\!]_{\phi,x=[\![u]\!]_\phi}$. And that if $t \equiv_{\beta\mathcal{R}} u$ then $\mathcal{M}_t = \mathcal{M}_u$ and $[\![t]\!]_\phi = [\![u]\!]_\phi$. We thus get the following theorem.

▶ **Theorem 16.** *Simple type theory is super-consistent.*

4.3 Simple type theory with a parametric quantifier

In a model of Simple type theory with a parametric quantifier, like in the previous section, we want to take $\mathcal{M}_o = \mathcal{B}$. But, unlike in the previous section, we do not have $o : Type$, but $o : type : Type$. So o is now an object.

In the previous section, we took $\mathcal{M}_t = \{e\}$ for all objects. This permitted to define $\mathcal{M}_{(t\ u)}$ and $\mathcal{M}_{\lambda x:C\ t}$ as $\mathcal{M}_t$ and validate β-reduction trivially. But this is not possible anymore in Simple type theory with a parametric quantifier, where $\mathcal{M}_o$ is $\mathcal{B}$ and $\mathcal{M}_{arrow(o,o)}$ is $\mathcal{F}(\mathcal{B}, \mathcal{B})$. So, we cannot define $\mathcal{M}_{\lambda x:type\ x}$ to be $\mathcal{M}_x$, but we need to define it as a function. To help to construct this function, we need to construct first another interpretation function $(\mathcal{N}_t)_t$ and parametrize the definition of $\mathcal{M}_t$ itself by a function ψ mapping variables of type A to elements of $\mathcal{N}_A$. Thus the model is a three layer model.

Like in the previous section, we want to define $\mathcal{M}_{\Pi x:C\ D,\psi}$, as the set of functions from $\mathcal{M}_{C,\psi}$ to $\mathcal{M}_{D,\psi'}$. But to define this set $\mathcal{M}_{D,\psi'}$, we need to extend the function ψ, mapping x to an element of $\mathcal{N}_C$. To have such an element of $\mathcal{N}_C$, we need to define $\mathcal{M}_{\Pi x:C\ D,\psi}$ as the set of functions mapping $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$ to an element of $\mathcal{M}_{D,(\psi,x=c')}$. As a consequence, if ϕ is a function mapping x of type A to some element of $\mathcal{M}_A$, we need to define $[\![(t\ u)]\!]_\phi$ not as $[\![t]\!]_\phi\ [\![u]\!]_\phi$ but as $[\![t]\!]_\phi\ \langle \mathcal{M}_{u,\psi}, [\![u]\!]_\phi\rangle$. As a consequence $[\![.]\!]$ must be parametrized by both ψ and ϕ.

Let $\mathcal{B} = \langle \mathcal{B}, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{B}), \tilde{\Pi}\rangle$ be a full Π-algebra.

Let $\{e\}$ be an arbitrary one-element set. Let $\mathcal{U}$ be a set containing $\mathcal{B}$ and $\{e\}$, and closed by function space and Cartesian product, that is such that if S and T are in $\mathcal{U}$ then so are $S \times T$ and $\mathcal{F}(S, T)$. Such a set can be constructed, with the replacement scheme, as follows

$$\begin{aligned} \mathcal{U}_0 &= \{\mathcal{B}, \{e\}\}, \\ \mathcal{U}_{n+1} &= \mathcal{U}_n \cup \{S \times T \mid S, T \in \mathcal{U}_n\} \cup \{\mathcal{F}(S, T) \mid S, T \in \mathcal{U}_n\}, \\ \mathcal{U} &= \bigcup_n \mathcal{U}_n. \end{aligned}$$

Then, let $\mathcal{V}$ be the smallest set containing $\{e\}$, $\mathcal{B}$, and $\mathcal{U}$, and closed by Cartesian product and dependent function space, that is, if S is in $\mathcal{V}$ and T is a family of elements of $\mathcal{V}$ indexed by S, then the set of functions mapping an element s of S to an element of T_s is an element of $\mathcal{V}$. As noted in [19], the construction of the set $\mathcal{V}$, unlike that of $\mathcal{U}$, requires an inaccessible cardinal. Note that $\mathcal{U}$ is both an element and a subset of $\mathcal{V}$.

▶ **Definition 17** (A model of Simple type theory with a parametric quantifier)**.**

- $\mathcal{N}_{Type} = \mathcal{N}_{Kind} = \mathcal{V}$,
- $\mathcal{N}_{\Pi x:C\ D}$ is the set $\mathcal{F}(\mathcal{N}_C, \mathcal{N}_D)$, except if $\mathcal{N}_D = \{e\}$, in which case $\mathcal{N}_{\Pi x:C\ D} = \{e\}$,
- $\mathcal{N}_{type} = \mathcal{U}$,
- $\mathcal{N}_\iota = \mathcal{N}_o = \mathcal{N}_{arrow} = \mathcal{N}_\Rightarrow = \mathcal{N}_\forall = \mathcal{N}_\eta = \mathcal{N}_\varepsilon = \{e\}$,
- $\mathcal{N}_x = \{e\}$,
- $\mathcal{N}_{\lambda x:C\ t} = \mathcal{N}_t$,
- $\mathcal{N}_{(t\ u)} = \mathcal{N}_t$.

- $\mathcal{M}_{Kind,\psi} = \mathcal{M}_{Type,\psi} = \mathcal{B}$,
- $\mathcal{M}_{\Pi x:C\ D,\psi,\phi}$ is the set of functions f mapping $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$ to an element of $\mathcal{M}_{D,(\psi,x=c')}$, except if for all c' in $\mathcal{N}_C$, $\mathcal{M}_{D,(\psi,x=c')} = \{e\}$, in which case $\mathcal{M}_{\Pi x:C\ D,\psi} = \{e\}$,
- $\mathcal{M}_{type,\psi} = \mathcal{B}$,
- $\mathcal{M}_{\eta,\psi}$ is the function of $\mathcal{F}(\mathcal{U}, \mathcal{V})$ mapping S to S,
- $\mathcal{M}_{\varepsilon,\psi}$ is the function of $\mathcal{F}(\{e\}, \mathcal{V})$, mapping e to $\{e\}$,
- $\mathcal{M}_{\iota,\psi} = \{e\}$,
- $\mathcal{M}_{o,\psi} = \mathcal{B}$,
- $\mathcal{M}_{arrow,\psi}$ is the function mapping S and T in $\mathcal{U}$ to the set $\mathcal{F}(\{e\} \times S, T)$, except if $T = \{e\}$ in which case it maps S and T to $\{e\}$,
- $\mathcal{M}_{\Rightarrow,\psi} = \mathcal{M}_{\forall,\psi} = e$,
- $\mathcal{M}_{x,\psi} = \psi x$,
- $\mathcal{M}_{\lambda x:C\ t,\psi}$ is the function mapping c in $\mathcal{N}_C$ to $\mathcal{M}_{t,(\psi,x=c)}$, except if for all c in $\mathcal{N}_C$, $\mathcal{M}_{t,(\psi,x=c)} = e$, in which case $\mathcal{M}_{\lambda x:C\ t,\psi} = e$,
- $\mathcal{M}_{(t\ u),\psi} = \mathcal{M}_{t,\psi}\ \mathcal{M}_{u,\psi}$, except if $\mathcal{M}_{t,\psi} = e$ in which case $\mathcal{M}_{(t\ u),\psi} = e$.

- $[\![Kind]\!]_{\psi,\phi} = [\![Type]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![\Pi x : C\ D]\!]_{\psi,\phi} = \tilde{\Pi}([\![C]\!]_{\psi,\phi}, \{[\![D]\!]_{(\psi,x=c'),(\phi,x=c)} \mid c' \in \mathcal{N}_C, c \in \mathcal{M}_{C,\psi}\})$,
- $[\![type]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![\iota]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![o]\!]_{\psi,\phi} = \tilde{\top}$,
- $[\![arrow]\!]_{\psi,\phi}$ is the function from $\mathcal{U} \times \mathcal{B}$ and $\mathcal{U} \times \mathcal{B}$ to $\mathcal{B}$ mapping $\langle S, a\rangle$ and $\langle T, b\rangle$ to $\tilde{\Pi}(a, \{b\})$,
- $[\![\Rightarrow]\!]_{\psi,\phi}$ is the function $\{e\} \times \mathcal{B}$ and $\{e\} \times \mathcal{B}$ to $\mathcal{B}$ mapping $\langle e, a\rangle$ and $\langle e, b\rangle$ to $\tilde{\Pi}(a, \{b\})$,
- $[\![\forall]\!]_{\psi,\phi}$ is the function mapping $\langle S, a\rangle$ in $\mathcal{U} \times \mathcal{B}$, and $\langle e, g\rangle$ in $\{e\} \times \mathcal{F}(\{e\} \times S, \mathcal{B})$ to $\tilde{\Pi}(a, \{(g\ \langle e, s\rangle) \mid s \in S\})$,
- $[\![\eta]\!]_{\psi,\phi}$ is the function from $\mathcal{U} \times \mathcal{B}$ to $\mathcal{B}$, mapping $\langle S, a\rangle$ to a,
- $[\![\varepsilon]\!]_{\psi,\phi}$ is the function from $\{e\} \times \mathcal{B}$ to $\mathcal{B}$, mapping $\langle e, a\rangle$ to a,
- $[\![x]\!]_{\psi,\phi} = \phi x$,
- $[\![\lambda x : C\ t]\!]_{\psi,\phi}$ is the function mapping $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$ to $[\![t]\!]_{(\psi,x=c'),(\phi,x=c)}$, except if for all $\langle c', c\rangle$ in $\mathcal{N}_C \times \mathcal{M}_{C,\psi}$, $[\![t]\!]_{(\psi,x=c'),(\phi,x=c)} = e$, in which case $[\![\lambda x : C\ t]\!]_{\psi,\phi} = e$,
- $[\![(t\ u)]\!]_{\psi,\phi} = [\![t]\!]_{\psi,\phi}\ \langle \mathcal{M}_{u,\psi}, [\![u]\!]_{\psi,\phi}\rangle$, except if $[\![t]\!]_{\psi,\phi} = e$, in which case $[\![(t\ u)]\!]_{\psi,\phi} = e$.

We prove that if $\Gamma \vdash t : B$ and ψ is a function mapping the variables $x : A$ of Γ to elements of $\mathcal{N}_A$, then $\mathcal{M}_{t,\psi} \in \mathcal{N}_B$. That if $\Gamma \vdash t : B$, ψ is a function mapping variables $x : A$ of Γ to elements of $\mathcal{N}_A$, and ϕ is a function mapping variables $x : A$ of Γ to elements of $\mathcal{M}_{A,\psi}$, then $[\![t]\!]_{\psi,\phi} \in \mathcal{M}_{B,\psi}$ and $[\![(u/x)t]\!]_{\psi,\phi} = [\![t]\!]_{\psi,x=\mathcal{M}_{u,\psi}\phi,x=[\![u]\!]_{\psi,\phi}}$. And that if $t \equiv_{\beta\mathcal{R}} u$ then $\mathcal{N}_t = \mathcal{N}_u$, $\mathcal{M}_{t,\psi} = \mathcal{M}_{u,\psi}$, and $[\![t]\!]_{\psi,\phi} = [\![u]\!]_{\psi,\phi}$. We thus get the following theorem.

▶ **Theorem 18.** *Simple type theory with a parametric quantifier is super-consistent.*

Note that the set $\mathcal{V}$, thus an inaccessible cardinal, are not really needed to prove the super-consistency of Simple type theory with a parametric quantifier if we can adapt the notion of model in such a way that the family $\mathcal{N}$ is defined for type families only. The systematic development of this notion of partial interpretation is left for future work.

A similar proof for the Calculus of constructions is given in the long version of the paper.

5 Termination of proof reduction

We finally prove that proof reduction terminates in the λΠ-calculus modulo any super-consistent theory.

In Deduction modulo theory, we can define a congruence with non terminating rewrite rules, without affecting the termination of proof reduction. For instance, the rewrite rule $c \longrightarrow c$ does not terminate, but the congruence it defines is the identity and proofs modulo this congruence are just proofs in pure Predicate logic. Thus, proof reduction in Deduction modulo this congruence terminates. So, in the λΠ-calculus modulo this congruence, the β-reduction terminates, but the $\beta\mathcal{R}$-reduction does not, as the $\mathcal{R}$-reduction alone does not terminate. Here, we restrict to prove the termination of β-reduction, not $\beta\mathcal{R}$-reduction. In some cases, like for the three theories presented above, the termination of the $\beta\mathcal{R}$-reduction is a simple corollary of the termination of the β-reduction. In some others, it is not.

The main notion used in this proof is that of *reducibility candidate* introduced by Girard [15]. Our inductive definition, however, follows that of Parigot [22].

▶ **Definition 19** (Candidates). The set $\tilde{\top}$ is defined as the set of strongly terminating terms.

Let C be a set of terms and S be a set of sets of terms. The set $\tilde{\Pi}(C, S)$ is defined as the set of strongly terminating terms t such that if $t \longrightarrow^*_\beta \lambda x : A\ t'$ then for all t'' in C, and for all D in S, $(t''/x)t' \in D$.

Candidates are inductively defined by the three rules

- the set $\tilde{\top}$ of all strongly terminating terms is a candidate,
- if C is a candidate and S is a set of candidates, then $\tilde{\Pi}(C, S)$ is a candidate,
- if S is a non empty set of candidates, then $\bigcap S$ is a candidate.

We write $\mathcal{C}$ for the set of candidates. The algebra $\langle \mathcal{C}, \leq, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{C}), \tilde{\Pi}\rangle$, where $\leq$ is the trivial relation such that $C \leq C'$ always, and $\tilde{\wedge}$ is any function from $\mathcal{C} \times \mathcal{C}$ to $\mathcal{C}$, for instance the constant function equal to $\tilde{\top}$, is a full Π-algebra. It is ordered by the subset relation and complete for this order. If C is a candidate, then all the elements of C strongly terminate.

Consider a super-consistent theory $\Sigma, \mathcal{R}$. We want to prove that β-reduction terminates in the λΠ-calculus modulo this theory.

As usual, we want to associate a candidate $[\![A]\!]$ to each term A in such a way that if t is a term of type A, then $t \in [\![A]\!]$. In the λΠ-calculus modulo theory, the main difficulty is to assign a candidates to terms in such a way that if $A \equiv B$ then $[\![A]\!] = [\![B]\!]$. For instance, if we have the rule $P \longrightarrow P \Rightarrow P$ that permits to type all lambda-terms, including non terminating ones, we should associate, to the term P, a candidate C such that $C = C \tilde{\Rightarrow} C$,

but there is no such candidate. For super-consistent theories, in contrast, such an assignment exists, as the theory has a model $\mathcal{M}$ valued in the Π-algebra $\langle \mathcal{C}, \leq, \tilde{\top}, \tilde{\wedge}, \mathcal{P}^+(\mathcal{C}), \tilde{\Pi} \rangle$. In this model, if t is a term of type B in some context Γ, then $[\![B]\!]_{\overline{\phi}}$ is a candidate.

We then prove that if Γ is a context, $\overline{\phi} = \phi_1, ..., \phi_n$ is be a sequence of valuations, σ is a substitution mapping every $x : A$ of Γ to an element of $[\![A]\!]_{\overline{\phi}}$ and t is a term of type B in Γ, then $\sigma t \in [\![B]\!]_{\overline{\phi}}$. We thus get the following theorem.

▶ **Theorem 20.** *In a super-consistent theory $\Sigma, \equiv$, all well-typed terms strongly terminate.*

Acknowledgements. The author wants to thank Frédéric Blanqui for very helpful remarks on a previous version of this paper.

References

1. A. Assaf, G. Burel, R. Cauderlier, D. Delahaye, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, and R. Saillard. Dedukti: a logical framework based on the lambda-Pi-calculus modulo theory. `http://www.lsv.ens-cachan.fr/~dowek/Publi/expressing.pdf`, 2016.
2. A. Assaf, G. Dowek, J.-P. Jouannaud, and J. Liu. Untyped confluence in dependent type theories. Submitted to publication, 2017.
3. A. Bauer, G. Gilbert, P. Haselwarter, M. Pretnar, and Ch. A. Stone. Design and implementation of the andromeda proof assistant. Types, 2016.
4. F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005.
5. A. Brunel, O. Hermant, and C. Houtmann. Orthogonality and boolean algebras for deduction modulo. In L. Ong, editor, *Typed Lambda Calculus and Applications*, volume 6690 of *Lecture Notes in Computer Science*, pages 76–90. Springer-Verlag, 2011.
6. H. Cirstea, L. Liquori, and B. Wack. Rewriting calculus with fixpoints: Untyped and first-order systems. In *Types*, volume 3085 of *Lectures Notes in Computer Science*. Springer-Verlag, 2003.
7. T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, pages 95–120, 1988.
8. D. Cousineau and G. Dowek. Embedding pure type systems in the lambda-pi-calculus modulo. In S. Ronchi Della Rocca, editor, *Typed lambda calculi and applications*, volume 4583 of *Lecture Notes in Computer Science*, pages 102–117. Springer-Verlag, 2007.
9. G. Dowek. Truth values algebras and proof normalization. In Th. Altenkirch and C. McBride, editors, *Types for proofs and programs*, volume 4502 of *Lecture Notes in Computer Science*, pages 110–124. Springer-Verlag, 2007.
10. G. Dowek, Th. Hardin, and C. Kirchner. Hol-lambda-sigma: an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11:1–25, 2001.
11. G. Dowek, Th. Hardin, and C. Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31:33–72, 2003.
12. G. Dowek and B. Werner. Proof normalization modulo. The Journal of Symbolic Logic, 68(4):1289–1316, 2003.
13. S. Foster and G. Struth. Integrating an automated theorem prover into agda. In M. Bobaru, K. Havelund, G.J. Holzmann, and R. Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*. Springer-Verlag, 2011.
14. H. Geuvers. A short and flexible proof of strong normalization for the calculus of constructions. In P. Dybjer, , B. Nordström, and J. Smith, editors, *Types for Proofs and Programs*, volume 996 of *Lecture Notes in Computer Science*, pages 14–38. Springer-Verlag, 1995.

15 J. Y. Girard. *Interprétation Fonctionnelle et Élimination des Coupures dans l'Arithmétique d'Ordre Supérieur*. PhD thesis, Université de Paris VII, 1972.

16 R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

17 P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

18 P.-A. Melliès and B. Werner. A generic normalisation proof for pure type systems. In E. Giménez and Ch. Paulin-Mohring, editors, *Types for Proofs and Programs*, volume 1512 of *Lecture Notes in Computer Science*, pages 254–276. Springer-Verlag, 1998.

19 A. Miquel and B. Werner. The not so simple proof-irrelevant model of CC. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs*, pages 240–258. Springer-Verlag, 2003.

20 Q. H. Nguyen, C. Kirchner, and H. Kirchner. External rewriting for skeptical proof assistants. *Journal of Automated Reasoning*, 29(309), 2002.

21 B. Nordström, K. Petersson, and J. M. Smith. Martin-löf's type theory. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 1–37. Clarendon Press, 2000.

22 M. Parigot. Proofs of strong normalization for second order classical natural deduction. In *Logic in Computer Science*, pages 39–46, 1993.

Proof Complexity Meets Algebra*

Albert Atserias[1] and Joanna Ochremiak[2]

1 **Universitat Politècnica de Catalunya, Barcelona, Spain**
`atserias@cs.upc.edu`
2 **Université Paris Diderot – Paris 7, Paris, France**
`joanna.ochremiak@gmail.com`

Abstract

We analyse how the standard reductions between constraint satisfaction problems affect their proof complexity. We show that, for the most studied propositional and semi-algebraic proof systems, the classical constructions of pp-interpretability, homomorphic equivalence and addition of constants to a core preserve the proof complexity of the CSP. As a result, for those proof systems, the classes of constraint languages for which small unsatisfiability certificates exist can be characterised algebraically. We illustrate our results by a gap theorem saying that a constraint language either has resolution refutations of bounded width, or does not have bounded-depth Frege refutations of subexponential size. The former holds exactly for the widely studied class of constraint languages of bounded width. This class is also known to coincide with the class of languages with Sums-of-Squares refutations of sublinear degree, a fact for which we provide an alternative proof. We hence ask for the existence of a natural proof system with good behaviour with respect to reductions and simultaneously small size refutations beyond bounded width. We give an example of such a proof system by showing that bounded-degree Lovász-Schrijver satisfies both requirements.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint Satisfaction Problem, Proof Complexity, Reductions, Gap Theorems

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.110

1 Introduction

The notion of efficient reduction is at the heart of all subareas of computational complexity. However, in some subareas such as proof complexity, even though the concept exists, it is much less developed. The study of the lengths of proofs has developed mostly by studying combinatorial statements, each somewhat in isolation. There is little theory, for instance, explaining why the best studied families of propositional tautologies are encodings of the pigeonhole principle or those derived from systems of linear equations over the 2-element field. Whether there is any connection between the two is an even less explored mystery.

Luckily this fact is subject to revision, especially if proof complexity exports its methods to the study of problems beyond universal combinatorial statements. Consider the NP-hard optimization problem called MAX-CUT. The objective is to find a partition of the vertices

* Both authors partially funded by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement ERC-2014-CoG 648276 (AUTAR). First author partially funded by MINECO through TIN2013-48031-C4-1-P (TASSAT2). Part of this work was done while the authors were in residence at the Simons Institute for the Theory of Computing.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 110; pp. 110:1–110:14

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

of a given graph which maximizes the number of edges that cross the partition. The best efficient approximation algorithm known for this problem relies on certifying a bound on the optimum of its semidefinite programming relaxation. Once the certificate for the relaxation is in place, a rounding procedure gives an approximate integral solution: at worst 87% of the optimum in this case [12].

In the example of the previous paragraph, the problem that is subject to proof complexity analysis is that of certifying a bound on the optimum of an arbitrary MAX-CUT instance. The celebrated Unique Games Conjecture (UGC) can be understood as a successful approach to explaining why current algorithms and proof complexity analyses stop being successful where they do, and reductions play an important role there [26]. One of the interesting open problems in this area is whether the analysis of the Sums-of-Squares semidefinite programming hierarchy of proof systems (SOS) could be used to improve over the 87% approximation ratio for MAX-CUT. Any improvement on this would improve the approximation status of all problems that reduce to it, and refute the UGC [16]. For the constraint satisfaction problem, in which all constraints must be satisfied, the analogue question was resolved very recently also by exploiting the theory of reducibility: in that arena, low-degree SOS unsatisfiability proofs exist only for problems of *bounded width* [11, 25].

The goal of this paper is to develop the standard theory of reductions between constraint satisfaction problems in a way that it applies to many of the proof systems from the literature, including but not limited to SOS. Doing this requires a good amount of tedious work, but at the same time has some surprises to offer that we discuss next.

Consider a constraint language B given by a finite domain of values, and relations over that domain. The instances of the constraint satisfaction problem (CSP) over B are given by a set of variables and a set of constraints, each of which binds some tuple of the variables to take values in one of the relations of B. The literature on CSPs has focussed on three different types of conditions that, if met by two constraint languages, give a reduction from the CSP of one language to the CSP of the other. These conditions are a) *pp-interpretability*, b) *homomorphic equivalence*, and c) *addition of constants to the core* (see [9, 5]). What makes these three types of reductions important is that they correspond to classical algebraic constructions at the level of the *algebras of polymorphisms* of the constraint languages. Indeed, pp-interpretations correspond to taking homomorphic images, subalgebras and powers. The other two types of reductions put together ensure that the algebra of the constraint language is idempotent. Thus, for any fixed algorithm, heuristic, or method $\mathcal{M}$ for deciding the satisfiability of CSPs, if the class of constraint languages that are solvable by $\mathcal{M}$ is closed under these notions of reducibility, then this class admits a purely algebraic characterization in terms of identities.

Our first result is that, for most proof systems P in the literature, each of these methods of reduction preserves the proof complexity of the problem with respect to proofs in P. Technically, what this means is that if B is obtained from B' by a finite number of constructions a), b) and c), then, for an appropriate fixed encoding scheme of the statement that an instance is unsatisfiable, efficient proofs of unsatisfiability in P for instances of B' translate into efficient proofs of unsatisfiability in P for instances of B. The propositional proof systems for which we prove this include DNF-resolution with terms of bounded size, bounded-depth Frege, and Frege. The semi-algebraic proof systems for which we prove it include Sherali-Adams, Lasserre/SOS, and Lovász-Schrijver of bounded and unbounded degree.

Our second main result is an application: we obtain unconditional gap theorems for the proof complexity of CSPs. Building on the bounded-width theorem for CSPs [4, 8], the

known correspondance between local consistency algorithms, existential pebble games and bounded width resolution [17, 2], the lower bounds for propositional and semi-algebraic proof systems [1, 19, 6, 7, 13, 10], and a modest amount of additional work to fill in the gaps, we prove the following strong gap theorem:

▶ **Theorem 1.** *Let B be a finite constraint language. Then exactly one of the following holds:*

1. *B has resolution refutations of bounded width and hence polynomial size,*
2. *B has neither Frege refutations of bounded depth and subexponential size, nor Lasserre/SOS refutations of sublinear degree.*

Moreover, case *1.* in Theorem 1 happens precisely if B has bounded width. As noted earlier, the collapse of Lasserre/SOS to bounded width was already known; here we give a different proof. As an immediate corollary we get that resolution is also captured by algebra, despite the fact that our methods fall short to prove that it is closed under reductions.

▶ **Corollary 2.** *Let B be a finite constraint language. Then B has resolution refutations of subexponential size if and only if B has resolution refutations of polynomial size, if and only if B has resolution refutations of bounded width.*

Our third main result is about proof systems that operate with polynomial inequalities beyond Lasserre/SOS. Theorem 1 raises a question of identifying a proof system that can surpass bounded width. In other words: is there a natural proof system for which the class of languages that have efficient unsatisfiability proofs is closed under the standard reducibility methods for CSPs, and that at the same time has efficient unsatisfiability proofs beyond bounded width? By the bounded-width theorem for CSPs, one way, and indeed the only way, of surpassing bounded width is by having efficient proofs of unsatisfiability for systems of linear equations over some finite Abelian group. In view of the limitations of certain semi-algebraic proof systems that are imposed by Theorem 1, it is perhaps a surprise that, as we show, bounded degree Lovász-Schrijver (LS) is such a proof system.

▶ **Theorem 3.** *Unsatisfiable systems of linear equations over the 2-element group have LS refutations of bounded degree and polynomial size.*

Proving this amounts to showing that Gaussian elimination over $\mathbb{Z}_2$ can be simulated by reasoning with low-degree polynomial inequalities over $\mathbb{R}$. The proof of this counter-intuitive fact relies on earlier work in proof complexity for reasoning about gaps of the type $(-\infty, c] \cup [c+1, +\infty)$, for $c \in \mathbb{Z}$, through quadratic polynomial inequalities [15].

We want to close by pointing out that another proof system that can efficiently solve CSPs of bounded width, and that at the same time goes beyond bounded width, is the proof system that operates with ordered binary decision diagrams from [3]. Although it looks unlikely that our methods could be used for this proof system, whether it is closed under the standard CSP reducibilities is something that was not checked, neither in [3], nor here.

2 Preliminaries

2.1 Propositional logic and proofs

A literal is a variable X or the negation of a variable $\overline{X}$. We think of $\wedge$ and $\vee$ as commutative, associative and idempotent. Negation is allowed only on literals, so formulas are in negation normal form. If A is a formula, we define its complement $\overline{A}$ by exchanging $\vee$ and $\wedge$ and negating literals. The size of a formula A is the number of symbols in it.

We work with a Tait-style proof system for propositional logic that we call *Frege*. Its rules are *axiom*, *cut*, *introduction of conjunction*, and *weakening*:

$$\frac{}{A \vee \overline{A}} \qquad \frac{C \vee A \quad D \vee \overline{A}}{C \vee D} \qquad \frac{C \vee A \quad D \vee B}{C \vee D \vee (A \wedge B)} \qquad \frac{C}{C \vee A}. \tag{1}$$

In these rules, C and D could be the empty formula 0 or its complement 1, and A is a formula. A *Frege proof* of A from a set of formulas H is a sequence of formulas ending with A each of which is either in H, or follows from previous formulas in the sequence by one of the inference rules. The proof is called a *refutation* of H if the last formula is the empty formula 0. As a proof system, Frege is sound and implicationally complete. If $\mathcal{C}$ is a class of formulas, a $\mathcal{C}$-Frege proof is one that has all its formulas in the class $\mathcal{C}$. The size of a proof is the sum of the sizes of the formulas in it.

A k-term is a conjunction of at most k literals and a k-clause is a disjunction of at most k literals. A k-DNF is a disjunction of k-terms and a k-CNF is a conjunction of k-clauses. We define the classes of $\Sigma_{t,k}$- and $\Pi_{t,k}$-formulas inductively. For $t = 1$, these are just the classes of k-DNF and k-CNF formulas, respectively. For $t \geq 2$, a formula is $\Sigma_{t,k}$ if it is a disjunction of $\Pi_{t-1,k}$-formulas, and it is $\Pi_{t,k}$ if it is a conjunction of $\Sigma_{t-1,k}$-formulas. We write Σ_t and Π_t for $\Sigma_{t,1}$ and $\Pi_{t,1}$, respectively. The t and the k in $\Sigma_{t,k}$ and $\Pi_{t,k}$ are called the *depth* and the *bottom fan-in*, respectively.

Observe that Σ_1-Frege is essentially resolution, and $\Sigma_{1,k}$-Frege is the system R(k) introduced by Krajicek [18], also known as Res(k), k-DNF resolution, and k-DNF Frege. This proof system is important for us because it is the weakest for which we can prove closure under reductions. It is a sound and implicationally complete proof system for proving k-DNFs from k-DNFs. A resolution proof has *width* k if all clauses in it are k-clauses.

In this paper, we use the expression *Frege proof of depth d and bottom fan-in k* to mean a $\Sigma_{d,k}$-Frege proof. *Bounded-depth Frege* means Σ_d-Frege for some d. This coincides with other definitions in the literature. Again, Frege of depth d and bottom fan-in k, as a proof system, is sound and implicationally complete for proving $\Sigma_{d,k}$-formulas from $\Sigma_{d,k}$-formulas.

2.2 Polynomials and algebraic proofs

Let $X_1, \ldots, X_n$ be n algebraic commuting variables ranging over $\mathbb{R}$. We define proof systems for inequalities $P \geq 0$, where P is a polynomial in $\mathbb{R}[X_1, \ldots, X_n]$. We think of equations $P = 0$ as two inequalities $P \geq 0$ and $-P \geq 0$. For our purposes it will suffice to have the variables range over $\{0, 1\}$. Accordingly, we introduce *twin* variables $\bar{X}_1, \ldots, \bar{X}_n$ with the meaning that $\bar{X}_i = 1 - X_i$ for $i = 1, \ldots, n$.

In all proof systems, the following axioms will be imposed on these variables:

$$X_i^2 - X_i = 0 \qquad \bar{X}_i^2 - \bar{X}_i = 0 \qquad X_i + \bar{X}_i - 1 = 0, \tag{2}$$

$$X_i \geq 0 \qquad \bar{X}_i \geq 0 \qquad 1 - X_i \geq 0 \qquad 1 - \bar{X}_i \geq 0 \qquad 1 \geq 0. \tag{3}$$

Observe that $X_i\bar{X}_i = 0$ follows from these axioms: multiply $X_i + \bar{X}_i - 1 = 0$ by X_i and use $X_i^2 - X_i = 0$. This sort of reasoning is captured by the proof systems we are about to define.

Let P and Q denote polynomials. In addition to the axioms in (2), we consider rules of inference for deriving polynomial inequalities: from $P \geq 0$ and $Q \geq 0$, derive $P + Q \geq 0$, and from $P \geq 0$ and $Q \geq 0$ derive $PQ \geq 0$. Also we allow square inequalities for free: $P^2 \geq 0$. These are called *addition*, *multiplication* and *positivity of squares*.

If H denotes a system of polynomial inequalities $P_1 \geq 0, \ldots, P_r \geq 0$, a *semi-algebraic proof* of $P \geq 0$ from H is a sequence of polynomial inequalities ending with $P \geq 0$ each of

which is either in H, or is an axiom inequality from (2) and (3), or follows from previous inequalities in the sequence by one of the inference rules. The semi-algebraic proof is called a *refutation* of H if the last inequality is $-1 \geq 0$. As a proof system for inequalities evaluated over $\{0,1\}$, this is sound and implicationally complete (we note, however, that without some restrictions on the domain of evaluation, completeness is not true).

The main complexity measures for semi-algebraic proofs are size and degree. Polynomials are typically represented as explicit sums of monomials, or as algebraic formulas or circuits. Using formulas or circuits as representations requires some additional technicalities that we want to avoid (see [22, 14]). For all our examples below, we use the representation of an explicit sum of monomials; its size includes the sizes of the coefficients.

The proofs in the *Lovász-Schrijver* (LS) proof system are semi-algebraic proofs for which the following restrictions apply: 1) the polynomial Q in the multiplication rule is either a positive real or a variable, and 2) the positivity-of-squares is not allowed. When it is allowed, the system is called *Positive Semidefinite Lovász-Schrijver* and is denoted LS$^+$. Originally the Lovász-Schrijver proof system was defined to manipulate quadratic polynomials only (see [21, 23]). We follow [15] and consider the extension to arbitrary degree. For LS- and LS$^+$-proofs, an important complexity measure originally studied by Lovász and Schrijver is its *rank*, which is the maximum nesting depth of multiplication by a variable in the proof. Note that, due to possible cancellations, the degree of an LS-proof could in principle be much smaller than its rank.

We define two additional proof systems called *Sherali-Adams* (SA) and *Lasserre/Sums-of-Squares* (SOS). One way to do that is by thinking of them as subsystems LS and LS$^+$ proof systems, respectively, with the additional restriction that all applications of the multiplication rule must *precede* all applications of the addition rule. Due to the structural restriction in which multiplications precede additions, we can think of a proof of $P \geq 0$ from H as a polynomial identity of the form

$$\sum_{i=1}^{r} c_i \cdot P_i \cdot \prod_{j \in J_i} X_j \prod_{k \in K_i} \bar{X}_k = P, \tag{4}$$

where $c_1, \ldots, c_r$ are non-negative real numbers, and $P_1, \ldots, P_r$ are polynomials such that either the inequality $P_i \geq 0$ is in the set of hypothesis H, or they are axiom polynomials from (2) and (3), or they are squares of polynomials, when these are allowed. Note that the size of an SA or SOS proof thought of as a semi-algebraic proof is polynomially related to the sum of the sizes of the non-zero c_i's in (4).

We close this section by noting the relationships between LS and SA proofs on one hand, and LS$^+$ and SOS proofs on the other. Clearly, each SA proof of degree d is also an LS proof of degree d. The converse is certainly not true, but what is true is that every LS proof of degree d and rank k can be converted into an SA proof of degree $d + k$, where the rank is the complexity measure for LS proofs that we defined earlier. The same relationships hold between LS$^+$ and SOS: every SOS-proof of degree d is an LS$^+$ of degree d, and every LS$^+$ proof of degree d and rank k can be converted into an SOS-proof of degree $d + k$. The conversions go by swapping the order in which the addition and the multiplication rules are applied in LS proofs, when they appear in the *wrong* order. See [20] for a related discussion.

2.3 Constraint satisfaction problem

There are many equivalent definitions of the constraint satisfaction problem. Here we use the definition in terms of homomorphisms. Below we introduce the necessary terminology.

A *relational vocabulary* L is a set of symbols, each symbol has an associated arity. A *structure* $\mathbb{B}$ over L is a set B, called a *domain* together with a set of relations over B. For each $R \in L$ or arity r, there is a relation $R(\mathbb{B}) \subseteq B^r$ sometimes called an *interpretation* of R in $\mathbb{B}$. We say that a relational structure is finite if its domain is finite and it has finitely many non-empty relations. For two structures $\mathbb{B}$ and $\mathbb{B}'$ over the same vocabulary L, a *homomorphism* from $\mathbb{B}$ to $\mathbb{B}'$ is a function $h\colon B \to B'$, which preserves all the relations, that is, if $(b_1, \ldots, b_r) \in R(\mathbb{B})$, then $(h(b_1), \ldots, h(b_r)) \in R(\mathbb{B}')$, for each $R \in L$.

For a fixed L-structure $\mathbb{B}$ over a relational vocabulary L, the constraint satisfaction problem of $\mathbb{B}$, denoted CSP($\mathbb{B}$), is the following computational problem: given a finite L-structure $\mathbb{A}$, decide whether there exists a homomorphism from $\mathbb{A}$ to $\mathbb{B}$. In the context of CSP the structure $\mathbb{B}$ is often called a *constraint language*. We usually assume that the constraint language $\mathbb{B}$ is finite.

To reason about propositional proof systems for CSP we use the following fixed encoding. By CNF($\mathbb{A}, \mathbb{B}$) we denote the CNF formula which has clauses

1. $\bigvee_{b \in B} X(a, b)$ for each $a \in A$,
2. $\overline{X(a, b_0)} \vee \overline{X(a, b_1)}$ for each $a \in A$ and $(b_0, b_1) \in B^2$ with $b_0 \neq b_1$,
3. $\overline{X(a_1, b_1)} \vee \cdots \vee \overline{X(a_r, b_r)}$ for each natural number r, each $R \in L$ of arity r, each $(a_1, \ldots, a_r) \in R(\mathbb{A})$, and each $(b_1, \ldots, b_r) \in B^r \setminus R(\mathbb{B})$.

It is not difficult to see that the formula CNF($\mathbb{A}, \mathbb{B}$) is satisfiable if and only if there is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$.

To reason about semi-algebraic proof systems in the context of CSP we use the following fixed encoding. By INEQ($\mathbb{A}, \mathbb{B}$) we denote the system of linear inequalities defined as follows:

1. $\sum_{b \in B} X(a, b) - 1 \geq 0$ for each $a \in A$,
2. $\bar{X}(a, b_0) + \bar{X}(a, b_1) - 1 \geq 0$ for each $a \in A$ and $(b_0, b_1) \in B^2$ with $b_0 \neq b_1$,
3. $\sum_{i=1}^{r} \bar{X}(a, b_i) - 1 \geq 0$ for each natural number r, each $R \in L$ of arity r, each $(a_1, \ldots, a_r) \in R(\mathbb{A})$, and each $(b_1, \ldots, b_r) \in B^r \setminus R(\mathbb{B})$.

It is easy to see that the above system of linear inequalities has a solution satisfying the axioms from (2) and (3) if and only if there is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$.

The existential k-pebble game is played on two relational structures $\mathbb{A}$ and $\mathbb{B}$ over the same vocabulary by two players called Spoiler and Duplicator. The players are given two corresponding sets of pebbles $\{a_1, \ldots, a_k\}$ and $\{b_1, \ldots, b_k\}$. In each round Spoiler picks one of the k pebbles $a_1, \ldots, a_k$, say a_i, and puts it on an element of the structure $\mathbb{A}$. Duplicator responds by picking the corresponding pebble b_i and placing it on some element of the structure $\mathbb{B}$. For simplicity, in any given configuration of the game let us identify a pebble with the element of the structure that it is placed on. Spoiler wins if at any point during the game the partial function $f : A \to B$ defined by $f(a_i) = b_i$, for each pebbled element a_i of $\mathbb{A}$, is either not well defined or is not a partial homomorphism. Otherwise, the Duplicator wins.

A finite relational structure $\mathbb{B}$ has *width* k if, for every finite structure $\mathbb{A}$ of the same vocabulary as $\mathbb{B}$, if there is no homomorphism from $\mathbb{A}$ to $\mathbb{B}$, then Spoiler wins the existential k-pebble game on $\mathbb{A}$ and $\mathbb{B}$. The structure $\mathbb{B}$ has *bounded width* if it has width k for some k. Structures of bounded width are exactly those structures for which CSP($\mathbb{B}$) can be solved by a local consistency algorithm [17].

3 Closure under reductions

There are three types of reductions often considered in the context of CSPs: a) pp-interpretability b) homomorphic equivalence c) addition of constants to a core.

Let $\mathbb{B}$ and $\mathbb{B}'$ be finite relational structures over finite vocabularies L and L'. The structure $\mathbb{B}'$ is *pp-definable* in $\mathbb{B}$ if is has the same domain and for every relation symbol $T \in L'$ the relation $T(\mathbb{B}')$ is definable in $\mathbb{B}$ by a *pp-formula*, i.e., a first order formula using only symbols from L, conjunction, equality, and existential quantification. Formally, for every relation symbol $T \in L'$ there exists a pp-formula $\phi_T(x_1, \ldots, x_r)$, where r is the arity of T, such that $T(\mathbb{B}') = \{(b_1, \ldots, b_r) \in B^r : \mathbb{B} \models \phi_T(x_1/b_1, \ldots, x_r/b_r)\}$.

Pp-interpretability is a generalization of pp-definability which allows for changing the domain of a CSP language. Given two relational structures $\mathbb{B}$ and $\mathbb{B}'$, we say that $\mathbb{B}'$ is *pp-interpretable* in $\mathbb{B}$ if there exist a positive integer n and a surjective partial function $f\colon B^n \to B'$ such that the preimages of all relations in $\mathbb{B}'$ (including the equality relation) and the domain of f are pp-definable in $\mathbb{B}$. One of the fundamental results of the *algebraic approach* to the constraint satisfaction problem is that, whenever $\mathbb{B}'$ is pp-interpretable in $\mathbb{B}$, the CSP of the language $\mathbb{B}'$ is not harder than the CSP of the language $\mathbb{B}$ [9].

Structures $\mathbb{B}$ and $\mathbb{B}'$ over a vocabulary L are *homomorphically equivalent* if there exist homomorphisms from $\mathbb{B}$ to $\mathbb{B}'$ and from $\mathbb{B}'$ to $\mathbb{B}$. Obviously, if L-structures $\mathbb{B}$ and $\mathbb{B}'$ are homomorphically equivalent, then any L-structure $\mathbb{A}$ maps homomorphically to $\mathbb{B}$ if and only if it maps homomorphically to $\mathbb{B}'$. So the CSP problems over both languages are the same.

Homomorphic equivalence allows us to focus on studying constraint satisfaction problems of well-behaved structures which in this context turn out to be those exhibiting little symmetry. A finite relational structure is called a *core* if all its endomorphisms are surjective. It is known that every relational structure has a homomorphically equivalent substructure that is a core. Core structures can be extended by one-element unary relations which we refer to as *constants*, without increasing the complexity of the language [9].

It has been shown recently [5] that any class of constraint languages that is closed under the constructions a), b) and c) has an algebraic characterization in terms of identities of height 1. Here we show that DNF Frege, bounded-depth Frege, Frege, Sherali-Adams, Sums-of-Squares and Lovász-Schrijver proof systems behave well with respect to those three types of reductions.

Let us fix relational structures $\mathbb{B}$ and $\mathbb{B}'$ such that $\mathbb{B}'$ is obtained from $\mathbb{B}$ by a finite sequence of constructions a), b) and c). There is a known polynomial-time computable transformation that maps instances $\mathbb{A}'$ of CSP($\mathbb{B}'$) to instances $\mathbb{A}$ of CSP($\mathbb{B}$) such that $\mathbb{A}$ is satisfiable if and only if $\mathbb{A}'$ is satisfiable, and the size of $\mathbb{A}$ is linear in the size of $\mathbb{A}'$. We prove that this transformation satisfies the following:

▶ **Theorem 4.** *For any positive integers t, k and s, and any instance $\mathbb{A}'$, if there is a Frege refutation of* CNF($\mathbb{A}, \mathbb{B}$) *of depth t, bottom fan-in k, and size s, then there is a Frege refutation of* CNF($\mathbb{A}', \mathbb{B}'$) *of depth t, bottom fan-in polynomial in k, and size polynomial in the size of $\mathbb{A}'$ and s.*

▶ **Theorem 5.** *For any positive integers k and s, and any instance $\mathbb{A}'$, if there is a Sherali-Adams, Sums-of-Squares or Lovász-Schrijver refutation of* INEQ($\mathbb{A}, \mathbb{B}$) *of degree k and size s, then there is, respectively, a Sherali-Adams, Sums-of-Squares or Lovász-Schrijver refutation of* INEQ($\mathbb{A}', \mathbb{B}'$) *of degree linear in k and size polynomial in the size of $\mathbb{A}'$ and s.*

We point out that Theorem 5 in the case of the Sherali-Adams and Sums-of-Squares proof systems can be extracted from [24] and [25]. We include it here to illustrate the broad applicability of the systematic proof-complexity approach.

The main idea in proving the above theorems for all the proof systems under consideration is the same. The refutation for an instance $\mathbb{A}$ of CSP($\mathbb{B}$) is transformed into a refutation for an instance $\mathbb{A}'$ of CSP($\mathbb{B}'$) by substituting the variables of CNF($\mathbb{A}, \mathbb{B}$) or INEQ($\mathbb{A}, \mathbb{B}$) by

DNFs with bounded terms and a bounded number of terms or by polynomials with bounded degree, a bounded number of monomials and coefficients from a fixed, finite set, respectively. The additional condition we need to ensure in order to control the growth of the size and depth/degree of the refutations is that each formula from $\mathrm{CNF}(\mathbb{A},\mathbb{B})$ and every polynomial inequality from $\mathrm{INEQ}(\mathbb{A},\mathbb{B})$ after applying the substitution is a logical consequence of a bounded number of formulas/inequalities from $\mathrm{CNF}(\mathbb{A}',\mathbb{B}')$ or $\mathrm{INEQ}(\mathbb{A}',\mathbb{B}')$, respectively.

4 Upper bound

We say that a finite relational structure $\mathbb{B}$ has *resolution refutations of bounded width* if there is a positive integer k such that, for every finite structure $\mathbb{A}$ over the same vocabulary, if there is no homomorphism from $\mathbb{A}$ to $\mathbb{B}$, then $\mathrm{CNF}(\mathbb{A},\mathbb{B})$ has a resolution refutation of width k. The goal of this section is to prove the following:

▶ **Theorem 6.** *Let $\mathbb{B}$ be a finite relational structure. The following are equivalent:*
1. *$\mathbb{B}$ has bounded width,*
2. *$\mathbb{B}$ has resolution refutations of bounded width.*

In preparation for the proof we revisit the characterization of resolution width in terms of existential pebble games from [2].

Let $L = \{R_0, \ldots, R_q\}$ be a finite relational vocabulary consisting of $q+1$ symbols of arity q. Let $\mathbb{S}_q$ be an L-structure with two-element domain $\{0,1\}$, where each relation $R_i(\mathbb{S}_q)$ encodes the set of valuations that satisfy a q-clause with i negated variables. More precisely, for $0 \leq i \leq q$, let $R_i(\mathbb{S}_q) = \{0,1\}^q \setminus \{(x_1,\ldots,x_q)\}$ where $(x_1,\ldots,x_q) \in \{0,1\}^q$ is the vector defined by $x_j = 0$ for $j > i$ and $x_j = 1$, otherwise. Now for every q-CNF F, we define an L-structure $\mathbb{A}_F$. Its domain is the set of variables in F, and $R_i(\mathbb{A}_F)$ is the set of all tuples $(X_1,\ldots,X_q)$ such that the clause $\overline{X_1} \vee \ldots \vee \overline{X_i} \vee X_{i+1} \vee \ldots \vee X_q$ belongs to F. We allow the variables in the clauses to repeat, so the definition covers clauses with less than q literals. Observe that partial homomorphisms from $\mathbb{A}_F$ to $\mathbb{S}_q$ correspond to partial truth assignments to the variables of F that do not falsify any clause from F. Hence, for every q-CNF F, it holds that F is satisfiable if and only if there is a homomorphism from $\mathbb{A}_F$ to $\mathbb{S}_q$.

▶ **Theorem 7** ([2])**.** *Let k and q be positive integers such that $k \geq q$ and let F be q-CNF. Then F has a resolution refutation of width k if and only if Spoiler wins the existential $(k+1)$-pebble game on $\mathbb{A}_F$ and $\mathbb{S}_q$.*

We use the above theorem to establish a similar correspondence between existential pebble games on structures $\mathbb{A}$ and $\mathbb{B}$ and bounded width resolution refutations of $\mathrm{CNF}(\mathbb{A},\mathbb{B})$.

▶ **Lemma 8.** *Let $\mathbb{A}$ and $\mathbb{B}$ be relational structures over the same vocabulary of maximum arity r, let $q = |B|$, and let k be an integer such that $k \geq q$ and $k \geq r$. Then:*
1. *if Spoiler wins the existential $(k+2)$-pebble game on $\mathbb{A}$ and $\mathbb{B}$, then $\mathrm{CNF}(\mathbb{A},\mathbb{B})$ has a resolution refutation of width $k+q$,*
2. *if Duplicator wins the existential $(k+2)$-pebble game on $\mathbb{A}$ and $\mathbb{B}$, then $\mathrm{CNF}(\mathbb{A},\mathbb{B})$ does not have a resolution refutation of width $k+1$.*

Proof of Theorem 6. For the implication 1 to 2, assume that $\mathbb{B}$ has bounded width, say l. Let $k = \max\{q, r, l\}$, where $q = |B|$ and r is the maximum arity of the vocabulary of $\mathbb{B}$. Let $\mathbb{A}$ be a structure over the same vocabulary and assume that there is no homomorphism from $\mathbb{A}$ to $\mathbb{B}$. Then Spoiler wins the existential l-pebble game on $\mathbb{A}$ and $\mathbb{B}$, and hence also the existential $(k+2)$-pebble game on $\mathbb{A}$ and $\mathbb{B}$, since $k+2 \geq l$. The hypotheses of Lemma 8

hold, so by part *1* CNF$(\mathbb{A},\mathbb{B})$ has a resolution refutation of width $k+q$. This shows that $\mathbb{B}$ has resolution refutations of width $k+q$, and hence resolution refutations of bounded width.

For the implication 2 to 1, assume that $\mathbb{B}$ has resolution refutations of width l. Again let $k=\max\{q,r,l\}$. Let $\mathbb{A}$ be a structure over the same vocabulary as $\mathbb{B}$ and assume that there is no homomorphism from $\mathbb{A}$ to $\mathbb{B}$. Then CNF$(\mathbb{A},\mathbb{B})$ has a resolution refutation of width l, and hence of width $k+1$ since $k+1\geq l$. The hypotheses of Lemma 8 hold, so by part *2* in that lemma, Spoiler wins the existential $(k+2)$-pebble game on $\mathbb{A}$ and $\mathbb{B}$. This shows that $\mathbb{B}$ has width $k+2$, and hence bounded width. ◀

5 Lower bounds

Let $d(n)$ and $s(n)$ be functions. We say that a finite relational structure $\mathbb{B}$ has Frege refutations of depth $d(n)$ and size $s(n)$ if, for every finite structure $\mathbb{A}$ over the same vocabulary, if there is no homomorphism from $\mathbb{A}$ to $\mathbb{B}$, then CNF$(\mathbb{A},\mathbb{B})$ has a Frege refutation of depth $d(|A|)$ and size $s(|A|)$. We say that $\mathbb{B}$ has Frege refutations of bounded depth and subexponential size if there exist $d(n)=O(1)$ and $s(n)=2^{n^{o(1)}}$ such that $\mathbb{B}$ has Frege refutations of depth $d(n)$ and size $s(n)$.

Similarly, we say that a finite relational structure $\mathbb{B}$ has Sums-of-Squares refutations of degree $d(n)$ if, for every finite structure $\mathbb{A}$ over the same vocabulary, if there is no homomorphism from $\mathbb{A}$ to $\mathbb{B}$, then INEQ$(\mathbb{A},\mathbb{B})$ has a Sums-of-Squares refutation of degree $d(|A|)$. We say that $\mathbb{B}$ has Sums-of-Squares refutations of sublinear degree if there exists $d(n)=o(n)$ such that $\mathbb{B}$ has Sums-of-Squares refutations of degree $d(n)$. We prove:

▶ **Theorem 9.** *Let $\mathbb{B}$ be a finite relational structure. The following are equivalent:*
1. *$\mathbb{B}$ has bounded width,*
2. *$\mathbb{B}$ has Frege refutations of bounded depth and subexponential size,*
3. *$\mathbb{B}$ has Sums-of-Squares refutations of sublinear degree.*

The equivalence of *1* and *3* is known [11, 25]. Here we provide an alternative proof. The implication *1* to *2* follows from Theorem 6: every resolution refutation is a Frege refutation of depth one, and if the refutation has bounded width, then it has polynomial size and hence subexponential size. The implication *1* to *3* follows from Theorem 6 via the fact that bounded-degree SA simulates bounded-width resolution: bounded-width resolution is simulated by bounded-degree SA, which implies Sums-of-Squares refutations of a constant, and hence sublinear, degree.

For both implications *2* to *1* and *3* to *1* we use an algebraic characterization of unbounded width. We begin with some definitions.

Let $G=(G,+,0)$ be a finite Abelian group. For each $g\in G$ and every $(z_1,\ldots,z_k)\in\mathbb{Z}^k$, we define a relation $R_{(g,z_1,\ldots,z_k)}=\{(g_1,\ldots,g_k)\in G^k : z_1g_1+\ldots+z_kg_k=g\}$, where z_ig_i is a shortcut for the sum of $|z_i|$ copies of $\mathrm{sign}(z_i)g_i$. Let $\sim$ be the equivalence relation on the set $G\times\mathbb{Z}^k$ that identifies tuples defining the same relation. Since there are only finitely many k-ary relations on the finite set G, the equivalence relation $\sim$ has finitely many equivalence classes. Let $L(G,k)$ be the relational vocabulary that for every equivalence class $[(g,z_1,\ldots,z_k)]$ has one k-ary relation symbol $E_{[(g,z_1,\ldots,z_k)]}$, and let $\mathbb{B}(G,k)$ be the $L(G,k)$-structure that has domain G and where each relation symbol $E_{[(g,z_1,\ldots,z_k)]}$ is interpreted as $R_{(g,z_1,\ldots,z_k)}$. The CSP of $\mathbb{B}(G,k)$ is called kLIN(G). Instances of kLIN(G) are systems of linear equations over the group G with k variables per equation.

▶ **Theorem 10** ([4, 8])**.** *Let $\mathbb{B}$ be a finite relational structure. The following are equivalent:*

1. $\mathbb{B}$ *does not have bounded width,*
2. *there exists a non-trivial finite Abelian group* G *such that* $\mathbb{B}(G,3)$ *is pp-interpretable in* $\mathbb{B}^+$*, where* $\mathbb{B}^+$ *is the expansion of the core of* $\mathbb{B}$ *with all constants.*

Thus, in view of Theorems 4 and 5, in order to prove that *2* implies *1*, and that *3* implies *1* in Theorem 9, it suffices to prove lower bounds for $3\mathrm{LIN}(G)$, for every non-trivial finite Abelian group G.

For bounded-depth Frege we appeal to the lower bound for the pigeonhole principle [1, 6, 19]. To use that we need to be able to encode the pigeonhole principle as an unsatisfiable system of equations over an arbitrary Abelian group G. In [7], such a reduction was obtained for the so-called *Tseitin formulas*, that encode a certain system of linear equations over $\mathbb{Z}_2$ that is derived from an expander graph. Here we adapt the formulas to encode systems of linear equations over arbitrary finite Abelian groups and then show that the reduction in [7] can be adapted to our formulas. For Sums-of-Squares, unlike for bounded-depth Frege, we do not need to adapt an existing lower bound proof from the literature for $\mathbb{Z}_2$ to all finite Abelian groups because this was already done. The lower bound that we need to complete the proof of Theorem 9 is the following.

▶ **Theorem 11** ([10]). *For every non-trivial finite Abelian group* G *there exists a positive* δ *such that for every sufficiently large integer* n *there is an* n*-variable unsatisfiable instance* $\mathbb{A}$ *of* $3\mathrm{LIN}(G)$ *such that every SOS refutation of* $\mathrm{INEQ}(\mathbb{A}, \mathbb{B}(G,3))$ *has degree at least* δn*.*

The exact statement from [10] is Theorem G.8 from Appendix G, which differs from the version above. However, the original one implies the variant that we need.

6 Upper bounds in Lovász-Schrijver

In this section we show that all unsatisfiable instances of $3\mathrm{LIN}(\mathbb{Z}_2)$ have LS refutation of degree 6 and size polynomial in the number of variables. Indeed, the argument to get polynomial-size upper bound in constant degree works equally well for $3\mathrm{LIN}(\mathbb{Z}_p)$, when p is prime, with some inessential complications. We focus on $\mathbb{Z}_2$ for simplicity.

6.1 Initial remarks on the encoding

We identify the elements of the two-element field $\mathbb{Z}_2$ with $\{0,1\}$. Let $\mathbb{E}$ be an instance of $k\mathrm{LIN}(\mathbb{Z}_2)$ with n variables. In the encoding $\mathrm{INEQ}(\mathbb{E}, \mathbb{B}(\mathbb{Z}_2,k))$ of $\mathbb{E}$ as a system of linear inequalities, there are four variables $X(a,0), X(a,1), \bar{X}(a,0), \bar{X}(a,1)$ for each variable a in $\mathbb{E}$. Note, however, that they are restricted to satisfy $X(a,0) = \bar{X}(a,1)$ and $\bar{X}(a,0) = X(a,1)$ by the inequality $X(a,0) + X(a,1) - 1 \geq 0$ from INEQ and the default equations in (2), which in this case read $X(a,0)^2 - X(a,0) = X(a,1)^2 - X(a,1) = 0$ and $X(a,0) + \bar{X}(a,0) - 1 = X(a,1) + \bar{X}(a,1) - 1 = 0$. Consequently, in the following we will ignore the variables of the type $X(a,0)$ and their twins and keep only the variables $X(a,1)$ and $\bar{X}(a,1)$. In order to simplify the notation even further, we will assume that the variables of $\mathbb{E}$ are called $X_1, \ldots, X_n$, and that those of INEQ are called $X_1, \ldots, X_n$ and $\bar{X}_1, \ldots, \bar{X}_n$.

We interpret the variables $X_1, \ldots, X_n$ as ranging over $\mathbb{Z}_2$ or $\mathbb{Q}$ depending on the context. Let E be an equation of $\mathbb{E}$, say $E : a_1X_1 + \cdots + a_nX_n = b$, where $a_1, \ldots, a_n \in \mathbb{Z}_2$ and $b \in \mathbb{Z}_2$. Without loss of generality we can assume that there are exactly k many a_i's that are 1. In INEQ, the encoding of this equation is given by the following inequalities:

$$\sum_{i \in T} \bar{X}_i + \sum_{i \in I \setminus T} X_i - 1 \geq 0 \qquad \text{for all } T \subseteq I \text{ such that } |T| \equiv 1 - b \mod 2,$$

where $I = \{i \in [n] : a_i \neq 0\}$. Note that $|I| = k$. We write $\mathcal{S}(E)$ to denote this set of inequalities; it has exactly 2^{k-1} many inequalities, and all of them are satisfied in $\mathbb{Q}$ by a $\{0,1\}$-assignment if and only if the equation E is satisfied in $\mathbb{Z}_2$ by the same assignment. Let $\mathcal{S}(\mathbb{E})$ be the union of all $\mathcal{S}(E)$ as E ranges over the equations in $\mathbb{E}$. Observe that, except for the small detail that only half of the variables are used, INEQ is basically the same as $\mathcal{S}(\mathbb{E})$.

6.2 Some technical lemmas

For every linear form $L(X_1, \ldots, X_n) = \sum_{i=1}^n a_i X_i$ with rational coefficients $a_1, \ldots, a_n$ and every integer c, let $D_c(L) = (L - c)(L - c + 1)$, which is a quadratic polynomial. In words, the inequality $D_c(L) \geq 0$ states that L does not fall in the open interval $(c - 1, c)$. Such statements have short proofs of low degree:

▶ **Lemma 12** ([15]). *For every integer c and for every linear form $L(X_1, \ldots, X_n) = \sum_{i=1}^n a_i X_i$ with integer coefficients $a_1, \ldots, a_n$, there is a LS proof of the inequality $D_c(L) \geq 0$ (from nothing) of degree 3 and size polynomial in $\max\{|a_i| : i = 1, \ldots, n\}$, $|c|$ and n.*

In the following, for $I \subseteq [n]$ and $T \subseteq I$, let $M_T^I(X_1, \ldots, X_n) := \prod_{i \in T} X_i \prod_{i \in I \setminus T} \bar{X}$. As usual, $M_\emptyset^I(X_1, \ldots, X_n) = 1$. Such polynomials are called *extended monomials*.

▶ **Lemma 13.** *For every $I \subseteq [n]$, there is an LS proof of $\sum_{T \subseteq I} M_T^I - 1 = 0$ (from nothing) of degree $|I|$ and size polynomial in $2^{|I|}$, and for every $T \subseteq I \subseteq [n]$, there is an LS proof of $\left(\sum_{i \in I} X_i - |T|\right) M_T^I = 0$ (from nothing) of degree $|I| + 1$ and size polynomial in $|I|$.*

6.3 Simulating Gaussian elimination

Now we prove the main result of this section.

▶ **Theorem 14.** *Let $\mathbb{E}$ be an instance of $3\mathrm{LIN}(\mathbb{Z}_2)$ with n variables and m equations. If $\mathbb{E}$ is unsatisfiable, then $\mathcal{S}(\mathbb{E})$ has an LS refutation of degree 6 and size polynomial in n and m.*

Proof. Write $\mathbb{E}$ in matrix form $AX = b$, where X is a column vector of n variables, A is a matrix in $\mathbb{Z}_2^{m \times n}$, and b is a vector in $\mathbb{Z}_2^m$. Let $a_{j,1}, \ldots, a_{j,n}$ be the j-th row of A, so the j-th equation of $\mathbb{E}$ is $E_j : a_{j,1}X_1 + \cdots + a_{j,n}X_n = b_j$. Assume $\mathbb{E}$ is unsatisfiable over $\mathbb{Z}_2$. Then b cannot be expressed as a $\mathbb{Z}_2$-linear combination of the columns of A, so the $\mathbb{Z}_2$-rank of the matrix $[\, A \mid b \,]$ exceeds the $\mathbb{Z}_2$-rank of A. Since the rank of A is at most n, this means that there exists a subset of at most n rows J such that, with arithmetic in $\mathbb{Z}_2$, we have $\sum_{j \in J} a_{j,i} = 0$ for every $i \in [n]$, and at the same time $\sum_{j \in J} b_j = 1$. In order to simplify the notation, we assume without loss of generality that $J = \{1, \ldots, |J|\}$.

For every $k \in \{0, \ldots, |J|\}$, define the linear form

$$L_k(X_1, \ldots, X_n) := \frac{1}{2}\left(\sum_{j=1}^{k}\sum_{i=1}^{n} a_{j,i}X_i + \sum_{j=k+1}^{|J|} b_j\right).$$

In this definition of L_k, the coefficients $a_{j,i}$ and b_j are interpreted as rationals. We provide proofs of $D_c(L_k) \geq 0$ for every $c \in R_k := \{0, \ldots, (k+1)n\}$ by reverse induction on $k \in \{0, \ldots, |J|\}$.

The base case $k = |J|$ is a special case of Lemma 12. To see why note that the condition $\sum_{j \in J} a_{j,i} = 0$ over $\mathbb{Z}_2$ means that, if arithmetic were done in $\mathbb{Q}$, then $\sum_{j \in J} a_{j,i}$ is an even natural number. But then all the coefficients of

$$L_{|J|}(X_1, \ldots, X_n) = \frac{1}{2}\sum_{j=1}^{|J|}\sum_{i=1}^{n} a_{j,i}X_i = \sum_{i=1}^{n}\left(\frac{1}{2}\sum_{j=1}^{|J|} a_{j,i}\right)X_i$$

are integers. Hence Lemma 12 applies.

Suppose now that $0 \leq k \leq |J| - 1$ and that we have a proof of $D_d(L_{k+1}) \geq 0$ available for every $d \in R_{k+1}$. Fix $c \in R_k$; our immediate goal is to give a proof of $D_c(L_k) \geq 0$. As k is fixed, write L in place of L_{k+1}, and let the $(k+1)$-st equation E_{k+1} be written as $\sum_{i \in I} X_i = b$, where $I = \{i \in [n] : a_{k+1,i} = 1\}$. Note that $L = L_k + \ell/2$ where $\ell := -b + \sum_{i \in I} X_i$. Fix $T \subseteq I$ such that $|T| \equiv b \bmod 2$, and let $d = c + (t - b)/2$ where $t = |T|$. Note that $d \in R_{k+1}$ as $c \in R_k$ and $0 \leq t \leq n$ and $0 \leq b \leq 1$ are such that $t - b$ is even. Multiplying $D_d(L) \geq 0$ by the extended monomial M_T^I we get $(L - d)(L - d + 1)M_T^I \geq 0$. Replacing $L = L_k + \ell/2$ in the factor $(L - d)$ and recalling $d = c + (t - b)/2$, this inequality can be written as

$$(L_k - c)(L - d + 1)M_T^I + (L - d + 1)\tfrac{1}{2}A \geq 0, \tag{5}$$

where $A := (\ell + b - t)M_T^I$. By the second part of Lemma 13 we have a proof of $A = 0$, and hence of $(L - d + 1)A/2 = 0$. Composing with (5) we get a proof of $(L_k - c)(L - d + 1)M_T^I \geq 0$. The same argument applied to the factor $(L - d + 1)$ of this inequality gives $(L_k - c)(L_k - c + 1)M_T^I \geq 0$. This is precisely $D_c(L_k)M_T^I \geq 0$. Adding up over all $T \subseteq I$ with $|T| \equiv b \bmod 2$ we get

$$D_c(L_k) \sum_{\substack{T \subseteq I \\ |T| \equiv b}} M_T^I \geq 0. \tag{6}$$

Now note that for each $T \subseteq I$ such that $|T| \equiv 1 - b \mod 2$, the inequality $-M_T^I \geq 0$ is the multiplicative encoding of one of the inequalities in $\mathcal{S}(E)$. Thus, it is not difficult to show that it has an SA derivation from this inequality of size polynomial in $|I|$ and degree $|I| + 1$. Therefore, we get proofs of $-M_T^I \geq 0$, and hence of $M_T^I = 0$, for every $T \subseteq I$ such that $|T| \equiv 1 - b \bmod 2$. But then also of $D_c(L_k)M_T^I = 0$ for every such T. Adding up and composing with (6) we get

$$D_c(L_k) \sum_{T \subseteq I} M_T^I \geq 0. \tag{7}$$

From Lemma 13 we get $1 - \sum_{T \subseteq I} M_T^I = 0$, and hence $D_c(L_k) - D_c(L_k) \sum_{T \subseteq I} M_T^I \geq 0$, from which $D_c(L_k) \geq 0$ follows from addition with (7).

At this point we proved $D_c(L_0) \geq 0$ for every $c \in R_0 = \{0, \ldots, n\}$. Recall now that $\sum_{j=1}^{|J|} b_j$ is odd, say $2q + 1$, and at most n. In particular $q + 1$ belongs to R_0 and $L_0 = q + 1/2$. Thus we have a proof of $D_{q+1}(L_0) \geq 0$ where $D_{q+1}(L_0) = -(1/2)(1/2) = -1/4$. Multiplying by 4 we get the contradiction $-1 \geq 0$. ◀

7 Conclusions and Open Questions

Theorems 4 and 5 imply that for the proof systems under consideration the class of constraint languages admitting efficient refutations can be characterised algebraically. For most of those proof systems such a characterisation follows from the fact that efficient proofs of unsatisfiability exist exactly for languages of bounded width. However, by Theorem 14 the class of constraint languages admitting efficient refutations in Lovász-Schrijver, and consequently also the class of constraint languages admitting efficient Frege refutations, exceed bounded width. At the same time both of those classes are shown to admit algebraic characterisations. Providing such characterisations is a natural open problem that arises from our work.

A related direction that is also suggested by our work is whether the proof complexity of approximating MAX CSPs is also preserved by reductions. On the one hand, it is known that pp-definability preserves *almost satisfiability*; i.e., if $\mathbb{B}'$ is pp-definable in $\mathbb{B}$, then if $\mathbb{A}'$ is

an instance of MAX CSP($\mathbb{B}'$) that is almost satisfiable, then its standard transformation into an instance $\mathbb{A}$ of MAX CSP($\mathbb{B}$) is also almost satisfiable. The question we raise is the following: For which proof systems is it also the case that if there are efficient proofs that $\mathbb{A}$ is far from satisfiable then there also are efficient proofs that $\mathbb{A}'$ is far from satisfiable? Depending on how the terms "almost satisfiable" and "far from satisfiable" are quantified, a positive answer for such questions could lead to an algebraic approach to the theory of approximability of MAX CSPs and the UGC.

References

1. M. Ajtai. The complexity of the pigeonhole principle. In *29th Annual IEEE Symposium on Foundations of Computer Science*, pages 346–355, 1988.
2. A. Atserias and V. Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, May 2008. A preliminary version appeared in CCC 2003. doi:10.1016/j.jcss.2007.06.025.
3. A. Atserias, Ph. G. Kolaitis, and M. Vardi. Constraint propagation as a proof system. In *10th International Conference on Principles and Practice of Constraint Programming*, volume 3258 of *Lecture Notes in Computer Science*, pages 77–91. Springer-Verlag, 2004.
4. L. Barto and M. Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, January 2014. doi:10.1145/2556646.
5. L. Barto, J. Opršal, and M. Pinsker. The wonderland of reflections. *CoRR*, abs/1510.04521, 2015. URL: http://arxiv.org/abs/1510.04521.
6. P. Beame, R. Impagliazzo, J. Krajícek, T. Pitassi, P. Pudlák, and A. Woods. Exponential lower bounds for the pigeonhole principle. In *24th Annual ACM Symposium on the Theory of Computing*, pages 200–220, 1992.
7. E. Ben-Sasson. Hard examples for bounded depth frege. In *34th Annual ACM Symposium on the Theory of Computing*, pages 563–572, 2002.
8. A. Bulatov. Bounded relational width. Manuscript, 2009.
9. A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.
10. Siu On Chan. Approximation resistance from pairwise-independent subgroups. *J. ACM*, 63(3):27:1–27:32, August 2016. doi:10.1145/2873054.
11. A. Dawar and P. Wang. Lasserre lower bounds and definability of semidefinite programming. *CoRR*, abs/1602.05409, 2016. URL: http://arxiv.org/abs/1602.05409.
12. M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995. doi:10.1145/227683.227684.
13. D. Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1–2):613–622, 2001.
14. D. Grigoriev and E. A. Hirsch. Algebraic proof systems over formulas. *Theoretical Computer Science*, 303(1):83 – 102, 2003.
15. D. Grigoriev, E. A. Hirsch, and D. V. Pasechnik. Complexity of semi-algebraic proofs. *Moscow Mathematical Journal*, 4(2):647–679, 2002.
16. S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007.
17. Ph. G. Kolaitis and M. Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 175–181. AAAI Press, 2000. URL: http://dl.acm.org/citation.cfm?id=647288.721266.

18 J. Krajícek. On the weak pigeonhole principle. *Fundamenta Mathematicæ*, 170(1–3):123–140, 2001.

19 J. Krajícek, P. Pudlák, and A. Woods. Exponential lower bound to the size of bounded depth Frege proofs of the pigeon hole principle. *Random Structures and Algorithms*, 7(1):15–39, 1995.

20 M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496, 2001.

21 L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

22 T. Pitassi. Algebraic propositional proof systems. In N. Immerman and Ph. G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 68–96. American Mathematical Society, 1997.

23 P. Pudlák. On the complexity of the propositional calculus. In *Sets and Proofs, Invited Papers from Logic Colloquium '97*, pages 197–218. Cambridge University Press, 1999.

24 J. Thapper and S. Živný. Sherali-adams relaxations for valued csps. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015*, pages 1058–1069, 2015.

25 J. Thapper and S. Živný. The limits of SDP relaxations for general-valued csps. *CoRR*, abs/1612.01147, 2016. URL: `http://arxiv.org/abs/1612.01147`.

26 M. Tulsiani. CSP gaps and reductions in the Lasserre hierarchy. In *41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 303–312, 2009.

A Circuit-Based Approach to Efficient Enumeration*†

Antoine Amarilli[1], Pierre Bourhis[2], Louis Jachiet[3], and Stefan Mengel[4]

1 **LTCI, Télécom ParisTech, Université Paris-Saclay, Paris, France**
antoine.amarilli@telecom-paristech.fr
2 **CRIStAL, CNRS UMR 9189 & Inria Lille, Lille, France**
pierre.bourhis@univ-lille1.fr
3 **Université Grenoble Alpes, Grenoble, France**
louis.jachiet@inria.fr
4 **CNRS, CRIL UMR 8188, Lens, France**
mengel@cril.fr

Abstract

We study the problem of enumerating the satisfying valuations of a circuit while bounding the *delay*, i.e., the time needed to compute each successive valuation. We focus on the class of *structured d-DNNF circuits* originally introduced in knowledge compilation, a sub-area of artificial intelligence. We propose an algorithm for these circuits that enumerates valuations with linear preprocessing and delay linear in the Hamming weight of each valuation. Moreover, valuations of constant Hamming weight can be enumerated with linear preprocessing and constant delay.

Our results yield a framework for efficient enumeration that applies to all problems whose solutions can be compiled to structured d-DNNFs. In particular, we use it to recapture classical results in database theory, for factorized database representations and for MSO evaluation. This gives an independent proof of constant-delay enumeration for MSO formulae with first-order free variables on bounded-treewidth structures.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases circuits, constant-delay, enumeration, d-DNNFs, MSO

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.111

1 Introduction

When a computational problem has many solutions, computing all of them at once can take too much time. *Enumeration algorithms* are an answer to this challenge, and have been studied in many contexts (see overview in [36]). They generally consist of two phases. First, in a *preprocessing phase*, the input is read and indexed. Second, in an *enumeration phase* that uses the preprocessing result, the solutions are computed one after the other. The goal is to limit the amount of time between each pair of successive solutions, which is called *delay*.

We focus on a well-studied class of efficient enumeration algorithms with very strict requirements: the preprocessing must be *linear* in the input size, and the delay between

* For the full version with proofs, see [3], https://arxiv.org/abs/1702.05589.
† This work was partly funded by the French ANR Aggreg project, by the CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, by the PEPS JCJC INS2I 2017 CODA, and by the Télécom ParisTech Research Chair on Big Data and Market Insights.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 111; pp. 111:1–111:15


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

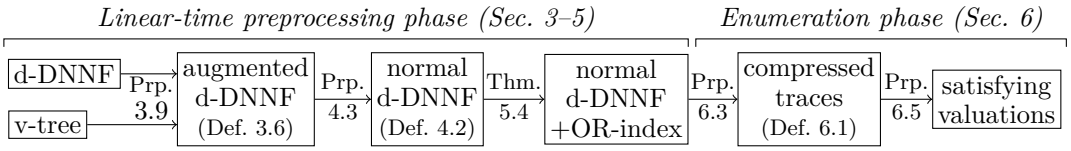


Figure 1 Overview of the proof of Theorem 2.1.

successive solutions must be *constant*. Such algorithms have been studied in particular for database applications, to enumerate query answers (see [19, 6, 20, 7, 8, 25, 24] and the recent survey [33]), or to enumerate the tuples of factorized database representations [29].

One shortcoming of these existing enumeration algorithms is that they are typically shown by building a custom index structure tailored to the problem, and designing ad hoc preprocessing and enumeration algorithms. This makes it hard to generalize them to other problems, or to implement them efficiently. In our opinion, it would be far better if enumeration for multiple problems could be done via one generic representation of the results to enumerate, reusing general algorithms for the preprocessing and enumeration phases.

This paper accordingly proposes a new framework for constant-delay enumeration algorithms, inspired by the field of *knowledge compilation* in artificial intelligence. Knowledge compilation studies how the solutions to computational problems can be *compiled* to generic representations, in particular classes of *Boolean circuits*, on which reasoning tasks can then be solved using general-purpose algorithms. In this paper, we show how this knowledge compilation approach can be implemented for constant-delay enumeration, by compiling to a prominent class of circuits from knowledge compilation called *deterministic decomposable negation normal form* (in short, d-DNNF) [17]. These circuits generalize several forms of branching programs such as OBDDs [18] and were recently shown to be more expressive than Boolean circuits of bounded treewidth [13]. Further, there are many efficient algorithms to compute d-DNNF representations of small width CNF formulae for a wide range of width notions [12], and even software implementations to compute such representations for given Boolean functions [30, 14]. d-DNNFs are also intimately related to state-of-the-art propositional model counters based on exhaustive DPLL [21], to syntactically multi-linear arithmetic circuits [32], and to probabilistic query evaluation in database theory [22].

Our main technical contribution is an efficient algorithm to enumerate the satisfying valuations of a d-DNNF under a standard structuredness assumption, namely, assuming that a so-called *v-tree* is given [31]: this assumption holds in all works cited above. Our first main result (Theorem 2.1) shows that we can enumerate the satisfying valuations of such a circuit with linear preprocessing and delay linear in the Hamming weight of each valuation. Further, our second main result (Theorem 2.2) shows that, if we impose a constant bound on the Hamming weight, we can enumerate the valuations with constant delay. In these results we express valuations succinctly as the set of the variables that they set to true.

To show our results, we consider d-DNNFs under a semantics with implicit negation: variables that are not tested must be set to zero. We call this semantics *zero-suppressed*, like zero-suppressed OBDDs [37]. Our preprocessing rewrites such d-DNNFs to a normal form (Section 4) and pre-computes a multitree reachability index on them (Section 5), allowing us to enumerate efficiently the *traces* of the circuit and the desired valuations (Section 6). To enumerate for d-DNNFs in standard semantics, we show how to rewrite them to zero-suppressed semantics, using the structuredness assumption and a new notion of *range gates* to make the process efficient (Section 3). The overall proof (see Figure 1) is very modular.

Our second contribution is to illustrate how our circuit-based framework and enumeration results can be useful in database theory. As a proof of concept, we present two known results

that we can extend, or recapture with an independent proof. First, we re-prove with our framework that the answers to MSO queries on trees and bounded-treewidth structures can be enumerated with linear preprocessing and delay linear in each assignment, i.e., constant-delay if the free variables are first-order. This was previously shown by Bagan [6] with a custom construction, by Kazana and Segoufin [25] using a powerful result of Colcombet [15], and by Courcelle [16] in a more general setting (but with $O(n \log n)$ preprocessing) using AND/OR-DAGs (that share some similarities with DNNFs). Our proof follows our proposed approach: we compute a circuit representation of the output following the provenance constructions in [4], and simply apply our enumeration result to this circuit. Second, we show how d-DNNFs generalize the deterministic factorized representations of relational instances studied in database theory [29]. We can thus enumerate with linear preprocessing and constant delay for arbitrary deterministic d-representations, which extends the result of [29].

The paper is structured as follows. Section 2 gives the main definitions and results. We then describe the preprocessing phase of our algorithm: we reduce the input circuit to zero-suppressed semantics in Section 3, rewrite it to a normal form in Section 4, and compute the multitree index in Section 5. We then describe the enumeration algorithm in Section 6. We present our two applications in Section 7 and conclude in Section 8. Due to space restrictions, many details and the proofs are found in the complete version [3].

2 Preliminaries and Problem Statement

Circuits. A circuit $C = (G, W, g_0, \mu)$ is a directed acyclic graph (G, W) whose vertices G are called *gates*, whose edges W are called *wires*, which has an *output gate* $g_0 \in G$, and where each gate $g \in G$ has a *type* $\mu(g)$ among $\wedge$ (AND-gate), $\vee$ (OR-gate), $\neg$ (NOT-gate), or var (variable). We represent the circuit with adjacency lists that indicate, for each gate $g \in G$, the gates having a wire to g (called the *inputs* of g), and the gates of which g is an input; the number of such gates is called respectively the *fan-in* and *fan-out* of g. The size $|C|$ of this representation is then $|G| + |W|$. We require that variables have fan-in zero, that NOT-gates have fan-in one, and we will always work on *negation normal form* (NNF) circuits where the input of NOT-gates is always a variable. A circuit without NOT-gates is called *monotone*.

We write C_{var} for the set of variables of C. A *valuation* of C_{var} is a function $\nu : C_{\text{var}} \to \{0, 1\}$. A circuit defines a *Boolean function* on C_{var}, i.e., a function ϕ that maps each valuation ν of C_{var} to $\{0, 1\}$. The image of ν by ϕ is defined by substituting each gate in C_{var} by its value in ν, evaluating the circuit using the standard semantics of Boolean operations, and returning the value of the output gate g_0. Note that AND-gates (resp., OR-gates) with no inputs always evaluate to 1 (resp., to 0) in this process. We call a gate *unsatisfiable* if it evaluates to 0 under all valuations (and *satisfiable* otherwise); we call it *0-valid* if it evaluates to 1 under the valuation which sets all variable gates to 0. We say that ν *satisfies* C if ϕ maps ν to 1 (i.e., g_0 evaluates to 1 under ν), and call ν a *satisfying valuation*.

For enumeration, we represent a valuation ν of C as the set S_ν of variables of C_{var} that it sets to 1, i.e., $\{g \in C_{\text{var}} \mid \nu(g) = 1\}$. We call S_ν an *assignment*, and a *satisfying assignment* if ν is a satisfying valuation. The *Hamming weight* $|\nu|$ of ν is the cardinality of S_ν. Unlike valuations, assignments of constant Hamming weight are of constant size, no matter the size of C_{var}. We write $\{\}$ for the empty assignment, and write $\emptyset$ for an empty set of assignments.

The main class of circuits that we will study are *d-DNNFs* [17], of which we now recall the definition. We say that an AND-gate g of a circuit C is *decomposable* if there is no pair $g_1 \neq g_2$ of input gates to g such that some variable $g' \in C_{\text{var}}$ has a directed path both to g_1 and to g_2: intuitively, a decomposable AND-gate is a conjunction of inputs on disjoint sets of

variables. We say that an OR-gate g of C is *deterministic* if there is no pair $g_1 \neq g_2$ of input gates of g and valuation ν of C such that g_1 and g_2 both evaluate to 1 under ν: intuitively, a deterministic OR-gate is a disjunction of mutually exclusive inputs. A circuit C is a *d-DNNF* if all its AND-gates are decomposable, and all its OR-gates are deterministic.

We further study the subclass of d-DNNFs called *structured d-DNNFs*, i.e., those having a *v-tree* [31]. A *v-tree* on a set S of variables is a rooted unranked ordered tree T whose set of leaves is exactly S. We write $<_T$ for the order on T in which the nodes are visited in a pre-order traversal. For a circuit C, we say that a v-tree T on the set C_{var} is a *v-tree* of C if there is a mapping λ from the gates of C to the nodes of T such that: (i) λ maps the variables of C to themselves; (ii) for each wire (g, g') of C, the node $\lambda(g)$ is a descendant of $\lambda(g')$ in T; and (iii) for each AND-gate g of C with inputs $g_1, \ldots, g_n$ (in this order), the nodes $\lambda(g_1), \ldots, \lambda(g_n)$ are descendants of $\lambda(g)$, none of them is a descendant of another, and we have $\lambda(g_1) <_T \cdots <_T \lambda(g_n)$. Note that having a v-tree implies (by iii) that all AND-gates are decomposable. A *structured d-DNNF* is a d-DNNF C given with a v-tree T of C.

Enumeration. As usual for efficient enumeration algorithms [33], we work in the RAM model with uniform cost measure (see, e.g., [2]), where pointers, numbers, labels for vertices and edges, etc., have constant size; thus an assignment has size linear in its Hamming weight.

An *enumeration algorithm with linear-time preprocessing* computes a set of results $\mathcal{S}(\mathcal{I})$ from an input instance $\mathcal{I}$. It consists of two parts. First, the *preprocessing phase* takes as input an instance $\mathcal{I}$ and produces in *linear time* an indexed instance $\mathcal{I}'$ and an initial *state*. Second, the *enumeration phase* repeatedly calls an algorithm $\mathcal{A}$. Each call to $\mathcal{A}$ takes as input the indexed instance $\mathcal{I}'$ and the current state, and returns a result and a new state: a special state value indicates that the enumeration is over so $\mathcal{A}$ should not be called again. The results produced by the calls to $\mathcal{A}$ must be exactly the elements of $\mathcal{S}(\mathcal{I})$, with no duplicates.

We say that the enumeration algorithm has *linear delay* if the time to produce each new output element $\mathcal{E}$ is linear in the size of $\mathcal{E}$ (and independent of the input instance $\mathcal{I}$). In particular, when the output elements have constant size, each element must be produced with constant delay, which we call *constant-delay enumeration*. The *memory usage* of an enumeration algorithm is the maximum number of memory cells used during the enumeration phase (not counting the indexed instance $\mathcal{I}'$, which resides in read-only memory), expressed as a function of the input instance size $|\mathcal{I}|$ and of the size $|\mathcal{O}|$ of the largest output (as in [6]). Note that constant delay does not imply a bound on memory usage, because the state can become large even if we only add a constant quantity of information at each step.

Main results. Our main theorem on circuit enumeration is the following:

▶ **Theorem 2.1.** *Given a structured d-DNNF C with a v-tree T, we can enumerate its satisfying assignments with linear-time preprocessing, linear delay, and memory usage $O(|\mathcal{O}| \cdot \log |C|)$, where $|\mathcal{O}|$ is the Hamming weight of the largest assignment.*

If we fix a maximal Hamming weight $k \in \mathbb{N}$, we can show constant-delay enumeration:

▶ **Theorem 2.2.** *For any $k \in \mathbb{N}$, given a structured d-DNNF C with a v-tree T, we can enumerate its satisfying assignments of Hamming weight $\leq k$ with preprocessing in time $O(|T| + k^2 \cdot |C|)$, delay in $O(k)$, and memory in $O(k \cdot \log |C|)$, i.e., linear-time preprocessing and constant delay for fixed k.*

In both results, remember that $|C|$ is the number of gates *and wires* of C. We prove our two results in Sections 3–6: the first three sections present the three steps of the linear-time

preprocessing algorithm, and the last one presents the enumeration algorithm. We then use the results for database applications in Section 7, in particular re-proving constant-delay enumeration for MSO queries with free first-order variables on bounded-treewidth structures.

The memory bound in our results is not constant and depends logarithmically on the input. While we think that this is reasonable, we also show constant-memory enumeration for some restricted circuit classes: the details are deferred to [3] for lack of space.

3 Reducing to Zero-Suppressed Semantics

We start our linear preprocessing by rewriting the input circuit to an alternative *zero-suppressed* semantics where negation is coded implicitly. For this rewriting, we will use the structuredness assumption on the circuit, in a weaker form called having a *compatible order*: this is the first thing that we present. We will also extend slightly our circuit formalism, to concisely represent sets of inputs with *range gates* that use this order: we present this second. Last, we present the alternative semantics, and give our translation result (Proposition 3.9).

Compatible orders. Our structuredness requirement is to have a *compatible order*:

▶ **Definition 3.1.** An *order* for a circuit C is a total order $<$ on C_{var}. For two variables $g_1, g_2 \in C_{\mathrm{var}}$, the *interval* $[g_1, g_2]$ consists of the variables g which are between g_1 and g_2 for $<$, i.e., $g_1 \leq g \leq g_2$ or $g_2 \leq g \leq g_1$. The *interval* of a gate g is then $[\min(g), \max(g)]$, where $\min(g)$ denotes the smallest gate according to $<$ that has a directed path to g, and $\max(g)$ is defined analogously. In particular, the interval of any $g \in C_{\mathrm{var}}$ is $[g, g] = \{g\}$.

We say that the order $<$ is *compatible* with C if, for every AND-gate g with inputs $g_1, \ldots, g_n$ (in this order), for all $1 \leq i < j \leq n$, we have $\max(g_i) < \min(g_j)$; in particular, the intervals of $g_1, \ldots, g_n$ are pairwise disjoint.

Note that, if a circuit C has compatible order $<$, every AND-gate g is decomposable: if some $g' \in C_{\mathrm{var}}$ had a directed path to two inputs of g then their intervals would intersect.

Observe further that, given a structured d-DNNF C with a v-tree T, we can easily compute a compatible order $<$ for C in linear time in T. Indeed, let $<$ be the restriction to C_{var} of the order $<_T$ on T given by pre-order traversal. Considering any suitable mapping λ from C to T, for any gate g, we know that $\min(g)$ is no less than the first leaf of T in $<$ reachable from $\lambda(g)$, and that $\max(g)$ is no greater than the last leaf reachable from $\lambda(g)$. The intervals of the inputs $g_1, \ldots, g_n$ to an AND-gate are then pairwise disjoint, because they are included in the sets of reachable leaves from the nodes $\lambda(g_1), \ldots, \lambda(g_n)$ in the v-tree, and none of these nodes is a descendant of another, so they cannot share any descendant leaf. Hence, if we know a v-tree T for C then we know an order $<$ for C.

Augmented circuits. We use compatible orders to define circuits with a new type of gates:

▶ **Definition 3.2.** For $k \in \mathbb{N}$, we define a *k-augmented circuit* C as a circuit with a compatible order $<$ and with k additional types of gates, called *range gates*: there are the *$=i$-range gates* for $0 \leq i < k$, and the *$\geq k$-range gates*. These gates must have exactly two inputs, which must be variables of C (they are not necessarily different, so we allow multi-edges in circuits for this purpose). We talk of *augmented circuits* when the value of k does not matter.

When evaluating a k-augmented circuit under a valuation ν, each $=i$-range gate g (resp., $\geq k$-range gate g) with inputs g_1 and g_2 evaluates to 1 if there are exactly i gates (resp., at least k gates) in $[g_1, g_2]$ set to 1 by ν; note that g may be unsatisfiable if $|[g_1, g_2]|$ is too small.

Range gates are related to the threshold gates studied in circuit complexity (see e.g. [11]), but we only apply them directly to variables. We can of course emulate range gates with standard gates, e.g., ≥ 0-gates always evaluate to 1, and a ≥ 1-range gate on g_1 and g_2 can be expressed as an OR-gate g having the interval $[g_1, g_2]$ as its set of inputs. However, the point of range gates is that we can now write this in constant space, thanks to $<$. This will be important to rewrite circuits in linear time to our alternative semantics.

Zero-suppressed semantics. We are now ready to introduce our alternative semantics for augmented circuits. We will do so only on *monotone* augmented circuits, i.e., without NOT-gates, because negation will be coded implicitly. We use the notion of *traces*:

▶ **Definition 3.3.** An *upward tree* T of a monotone augmented circuit $C = (G, W, \mu, g_0)$ is a subgraph (G', W') of C, with $G' \subseteq G$ and $W' \subseteq W$, which is a rooted tree up to reversing the direction of the wires. For all $(g', g) \in W'$, we call $g' \in G'$ a *child* of $g \in G'$ in T, and call g the *parent* of g' in T; note that g' is an input of g in C. A gate $g \in G'$ in T is an *internal gate* of T if it has a child in T, and a *leaf* otherwise. T is a *partial trace* if its internal gates are AND-gates and OR-gates and if its gates satisfy the following:

- for every AND-gate g in T, *all* its inputs in C are children of g in T;
- for every OR-gate g in T, *exactly one* of its inputs in C is a child of g in T.

Note that T cannot contain OR-gates with no inputs, and that its leaves consist of range gates, variable gates, and AND-gates with no inputs. We call T a *trace* of C if its root is g_0.

We define traces as trees, not general DAGs, because we cannot reach the same gate in a trace by two different paths (remember that AND-gates in augmented circuits are decomposable). We can see each trace (G', W') of $C = (G, W, \mu, g_0)$ as an augmented circuit (G', W', μ, g_0), up to adding to range gates in the trace their inputs in C, and we then have:

▶ **Observation 3.4.** *A valuation ν of a monotone augmented circuit C satisfies C if and only if ν satisfies a trace of C.*

Observe that we can check if a valuation ν of C satisfies a trace T simply by looking at the value of ν on the leaves of T; the definition of ν outside the intervals of the leaves does not matter. We will change this point to define zero-suppressed semantics, where ν can only satisfy T if it maps to 0 all the other variables. We then call ν a *minimal valuation* of T:

▶ **Definition 3.5.** Let C be a monotone augmented circuit, ν be a valuation of C, and T be a trace or partial trace of C. We call ν a *minimal valuation* of T if:

- For every variable g in T, we have $\nu(g) = 1$;
- For every $\bowtie i$-range gate g in T with inputs g_1 and g_2 in C (where $\bowtie \in \{=, \geq\}$ and $i \in \mathbb{N}$), the number n of variables in $[g_1, g_2]$ that are set to 1 by ν satisfies the constraint $n \bowtie i$;
- All other variables of C_{var} are set to 0 by ν.

Note that this implies that ν satisfies T. We call ν a *minimal valuation* for a gate g of C (resp., for C) if it is a minimal valuation of a partial trace rooted at g (resp., at the output g_0).

Note that C may have two minimal valuations ν_1 and ν_2 whose assignments S_1 and S_2 are such that $S_1 \subsetneq S_2$ (see, e.g., Example 3.7 below). Minimality only imposes that, relatively to a trace T, the valuation sets to 0 all variables that are not tested in T. Minimal valuations allow us to define the *zero-suppressed semantics* of a monotone augmented circuit C: the satisfying valuations of C in this semantics are those that are minimal for some trace.

▶ **Definition 3.6.** A monotone augmented circuit C in *zero-suppressed semantics* captures the (generally non-monotone) Boolean function Φ mapping a valuation ν to 1 iff ν is a minimal valuation for C. We call $S(C)$ the set of satisfying assignments of C in this semantics.

We call C a *d-DNNF in zero-suppressed semantics* if it satisfies the analogue of determinism: there is no OR-gate g with two inputs $g_1 \neq g_2$ and valuation ν of C that is a minimal valuation for both g_1 and g_2. (Decomposability again follows from the compatible order.)

▶ **Example 3.7.** Consider the monotone circuit C whose output gate is an OR-gate with three inputs: x, y, and an AND-gate of y and z. The circuit C captures $x \vee y$ in standard semantics, and it is not a d-DNNF. C has three traces, having one minimal valuation each. In the zero-suppressed semantics, we have $S(C) = \{\{x\}, \{y\}, \{y, z\}\}$, and C captures the Boolean function $(x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge y)$. Further, C is a d-DNNF in that semantics.

Zero-suppressed semantics makes enumeration easier, because it expresses negation implicitly in a very concise way. The name is inspired by zero-suppressed OBDDs [37, Chapter 8]: variables that are not tested when following a trace are implicitly set to 0. We can equivalently define the assignments $S(C)$ of C inductively as follows:

▶ **Lemma 3.8.** *Let C be a monotone augmented circuit. Let us define inductively a set of assignments $S(g)$ for each gate g in the following way:*

- *for all $g \in C_{\mathrm{var}}$, we set $S(g) := \{g\}$;*
- *for all $\bowtie i$-range gates g with inputs g_1 and g_2, we set $S(g) := \{t \subseteq [g_1, g_2] \mid |t| \bowtie i\}$;*
- *for all OR-gates g with inputs $g_1, \ldots, g_n$, we set $S(g) := \bigcup_{1 \leq i \leq n} S(g_i)$ (with $S(g) = \emptyset$ if g has no inputs);*
- *for all AND-gates g with inputs $g_1, \ldots, g_n$, we set $S(g) := \{S_1 \cup \cdots \cup S_n \mid (S_1, \ldots, S_n) \in \prod_{1 \leq i \leq n} S(g_i)\}$ (with $S(g) = \{\{\}\}$ if g has no inputs); observe that the unions are always disjoint because C has a compatible order.*

Then, for any gate g, the set $S(g)$ contains exactly the assignments that describe a minimal valuation for g. In particular, for g_0 the output gate of C, the set $S(g_0)$ is exactly $S(C)$.

We can now state our main reduction result for this section: we can rewrite any d-DNNF to an equivalent d-DNNF in zero-suppressed semantics, by introducing ≥ 0-range gates to write explicitly that the variables not tested in a trace are unconstrained:

▶ **Proposition 3.9.** *Given a d-DNNF circuit C and a compatible order $<$, we can compute in linear time a monotone 0-augmented circuit C^* having $<$ as a compatible order, such that C^* is a d-DNNF in zero-suppressed semantics and such that $S(C^*)$ is exactly the set of satisfying assignments of C.*

4 Reducing to Normal Form Circuits

In this section, given Proposition 3.9, we work on a monotone 0-augmented d-DNNF circuit C in zero-suppressed semantics, with a compatible order $<$ to define the semantics of range gates. We present our next two preprocessing steps for the enumeration of the assignments $S(C)$ of C: restricting our attention to valuations of the right Hamming weight (for Theorem 2.2 only), and bringing C to a normal form that makes enumeration easier.

Homogenization. Our input augmented circuit C in zero-suppressed semantics may have satisfying assignments of arbitrary Hamming weight. When proving Theorem 2.1, this is intended, and the construction that we are about to describe is not necessary. However, when proving Theorem 2.2 about enumerating valuations of constant weight, we need to restrict

our attention to such valuations, to ensure constant delay. We do so using the following homogenization result, adapted from the technique of Strassen [34]:

▶ **Proposition 4.1.** *Given $k \in \mathbb{N}$ and a monotone augmented d-DNNF circuit C in zero-suppressed semantics with compatible order $<$, we can construct in time $O(k^2 \cdot |C|)$ a monotone augmented d-DNNF circuit C' in zero-suppressed semantics with compatible order $<$ such that $S(C') = \{t \in S(C) \mid |t| \leq k\}$.*

Proof Sketch. We create $k+2$ copies of each gate g, with each copy capturing the assignments of a specific weight from 0 to k inclusive (or, for the $k+2$-th copy, the assignments with weight $> k$). In particular, for $\geq$0-gates g, for $0 \leq i \leq k$, we use an $=i$-gate for the copy of g capturing weight i. We then re-wire the circuit so that weights are correctly preserved. ◀

Note that this is the only place where our preprocessing depends on k: in particular, for constant k, the construction is linear-time. This result allows us to assume in the sequel that the set of assignments of the circuit in zero-suppressed semantics contains precisely the valuations that we are interested in, i.e., those that have suitable Hamming weight.

Normal form. Now that we have focused on the interesting valuations of our circuit C, we can bring it to our desired normal form:

▶ **Definition 4.2.** A *normal* circuit C is a monotone augmented circuit such that:

- C is *arity-two*, i.e., each gate has fan-in at most two.
- C is $\emptyset$-*pruned*, i.e., no gate g is unsatisfiable (i.e., each gate has some minimal valuation).
- C is $\{\}$-*pruned*, i.e., no gate g is 0-valid (i.e., the valuation that sets all variables to 0 is not a minimal valuation for any gate).
- C is *collapsed*, i.e., it has no AND-gate with fan-in 1.
- C is *discriminative*, i.e., for every OR-gate g with an input that is not an OR-gate (we call g an *exit*), g has fan-in 1, fan-out 1, and the one gate with g as input is an OR-gate.

C is a *normal d-DNNF* if it is additionally a d-DNNF in the zero-suppressed semantics.

The pruned requirements slightly weaken the expressiveness of normal circuits C, because they forbid that $S(C) = \emptyset$ or $\{\} \in S(C)$, which are easy to handle separately. We then have:

▶ **Proposition 4.3.** *Given a monotone augmented d-DNNF circuit C in zero-suppressed semantics with compatible order $<$ and with $S(C) \neq \emptyset$ and $S(C) \neq \{\{\}\}$, we can build in $O(|C|)$ a normal d-DNNF C', with $<$ as a compatible order, such that $S(C') = S(C) \backslash \{\{\}\}$.*

Proof Sketch. We reuse the construction of Proposition 4.1 with $k = 1$ to split the gates so that they are not 0-valid, eliminate bottom-up the unsatisfiable gates, make C arity-two in a straightforward way, collapse all AND-gates with fan-in 1, and make C discriminative by inserting new OR-gates (i.e., the exits) on all wires from non-OR-gates to OR-gates. ◀

This result allows us to assume in the sequel that we are working with normal d-DNNFs.

5 Indexing OR-Components

This section presents the last step of our preprocessing. Remember that we now work with a normal d-DNNF, and we want to enumerate its set of assignments. Intuitively, this last preprocessing will help us to enumerate the choices that can be made at OR-gates. Formally, we will work on the *OR-components* of our circuit:

▶ **Definition 5.1.** The *OR-component* K of an OR-gate g in a normal circuit C is the set of OR-gates that can be reached from g by going only through OR-gates, following wires in either direction. We abuse notation and also see K as a DAG, whose vertices are the gates of K, and whose edges are the wires between them.

Recall from Definition 4.2 that, as C is discriminative, all gates of an OR-component K with no inputs in K must be exits; we call them the *exits* of K. For a gate g in K, the *exits* of g are the gates of K that have a directed path to g in K; intuitively, they are the "possible choices" for a partial trace rooted at g. Our goal is to preprocess each OR-component of C to be able to enumerate efficiently the exits of all OR-gates of C. This enumeration task is tricky, however: exploring K naively when enumerating would take time dependent of C, but materializing a reachability index would take quadratic preprocessing time. Thus, we design an efficient indexing scheme, using the fact that OR-components are *multitrees*:

▶ **Definition 5.2.** A DAG G is a *multitree* if it has no pair $n \neq n'$ of vertices such that there are two different directed paths from n to n'. In particular, forests are multitrees, and so are polytrees (DAGs with no undirected cycles).

▶ **Lemma 5.3.** *For any normal d-DNNF C, each OR-component of C is a multitree.*

We can then prepare the enumeration of exits of gates in OR-components, by designing an efficient and generic indexing scheme on *multitrees* (see [3]). We deduce:

▶ **Theorem 5.4.** *Given a normal d-DNNF C, we can compute in $O(|C|)$ a structure called* OR-index *allowing us to do the following: given an OR-gate g of C, enumerate the exits of g in its OR-component K, with constant delay and memory usage $O(\log |K|)$.*

6 Enumerating Assignments

We have described in the previous sections our linear-time preprocessing on the input circuit: this produces a normal d-DNNF C together with an OR-index, and we wish to enumerate its assignments $S(C)$ in zero-suppressed semantics. In this section, we show that we can enumerate the elements of $S(C)$, producing each assignment t with delay $O(|t|)$.

To prove this, we will go back to our definition of zero-suppressed semantics in Section 3, namely, the minimal valuations of the traces of C (recall Definition 3.3). We will proceed in two steps. First, we use our preprocessing and the OR-index to show an efficient enumeration scheme for the traces of C, in a compact representation called *compressed traces*. Second, we show how to enumerate efficiently the minimal valuations of a compressed trace.

Compressed traces. We cannot enumerate traces directly because they can be arbitrarily large (e.g., contain long paths of OR-gates) even for assignments of small weight. We accordingly define *compressed traces* as a variant of traces that collapse such paths:

▶ **Definition 6.1.** An *OR-path* of a monotone augmented circuit $C = (G, W, \mu, g_0)$ is a path from $g \in G$ to $g' \in G$ where all intermediate gates are OR-gates; in particular if $(g, g') \in W$ then there is an OR-path from g to g'. A *compressed upward tree* of C is a pair (G', W') where $G' \subseteq G$ and where $W' \subseteq G' \times G'$ is such that for each $(g, g') \in W'$ there is an OR-path from g to g': we require that T is a rooted tree up to reversing the direction of the edges. T is a *compressed partial trace* if its internal gates are AND-gates and OR-gates such that:

- for every AND-gate g in T, *all* its inputs in C are children of g in T;
- for every exit g in T (it is an OR-gate), its one input in C is a child of g in T;
- for every non-exit OR-gate g in T, *exactly one* of its *exits* g' in C is a child of g in T.

We write $|T| := |G'|$. We call T a *compressed trace* of C if its root is g_0. The *minimal valuations* of a compressed trace are defined like for non-compressed traces (Definition 3.5).

The use of compressed traces is that their size is linear in that of their minimal valuations:

▶ **Lemma 6.2.** *For any compressed trace T of a normal circuit C and minimal valuation ν for T and C, we have $|T| \leq 6 \cdot |\nu|$.*

From a trace T in a normal d-DNNF C, we can clearly define a compressed trace T' with the same leaves, as follows. Whenever T contains an OR-gate g whose parent gate g' in T is not an OR-gate (or when g is the root of T), as g cannot be an exit, we know that there is a OR-path in T from g to an exit g'' of g in its OR-component. We "compress" this OR-path in T' as an edge from g to g''. Conversely, given a compressed trace T', we can fill it to a trace T with the same leaves, by replacing each edge from g to g' by a witnessing OR-path; there is only one way to do so because OR-components are multitrees (Lemma 5.3). Hence, there is a bijection between traces and compressed traces that preserves the set of leaves. As the minimal valuations of traces and compressed traces are defined in the same way from this set, we can simply enumerate compressed traces instead of traces. We can then show:

▶ **Proposition 6.3.** *Given a normal d-DNNF C with its OR-index, we can enumerate its compressed traces, with the delay to produce each compressed trace T being in $O(|T|)$.*

In particular, if all compressed traces have constant size, then the delay is constant.

Proof Sketch. At each AND-gate, we enumerate the lexicographic product of the partial traces of its two children; at each OR-gate, we enumerate its exits using the OR-index. ◀

Enumerating valuations of a compressed trace. We now show how, given a compressed trace T, we can enumerate its minimal valuations (recall Definition 3.5). Restricting our attention to the leaves of T, we can rephrase our problem in the following way:

▶ **Definition 6.4.** The *assignment enumeration problem* for a total order $<$ on gates C_{var} is as follows: given pairwise disjoint intervals $[g_1^-, g_1^+], \ldots, [g_n^-, g_n^+]$, and cardinality constraints $\bowtie_1 i_i, \ldots, \bowtie_n i_n$, where $0 < i_j \leq \left|[g_j^-, g_j^+]\right|$ and $\bowtie_j \in \{=, \geq\}$, enumerate the values of the products $t_1 \times \cdots \times t_n$ for all the assignments of the $t_j \subseteq [g_j^-, g_j^+]$ such that $|t_j| \bowtie_j i_j$ for all j.

Indeed, remember that, as C is $\{\}$-pruned, the leaves of T consist of variables and range gates, and their intervals are pairwise disjoint thanks to decomposability. A $\bowtie i$-gate with inputs g^-, g^+ codes the interval $[g^-, g^+]$ with cardinality constraint $\bowtie i$, and a variable g simply codes $[g, g]$ with constraint $= 1$. Further, thanks to $\{\}$-pruning, we know that no range gate is labeled with $= 0$ or ≥ 0, and thanks to $\emptyset$-pruning, we know that no range gate is labeled with an infeasible cardinality constraint. We claim:

▶ **Proposition 6.5.** *We can enumerate the solutions to the assignment enumeration problem for $<$ on C_{var}, with each solution t being produced with delay linear in its size $|t|$.*

Again, this is constant-delay when all solutions have size bounded by a constant.

Proof Sketch. We enumerate the possible assignments of weights to intervals with constant-delay, to reduce to the case where all cardinality constraints are equalities. We then enumerate the assignments in lexicographic order, using an existing scheme [26, Section 7.2.1.3]. ◀

We have concluded the proof of Theorem 2.1 (see Figure 1) and 2.2. Given our input d-DNNF C and v-tree T rewritten to a compatible order, we rewrite C to an equivalent normal d-DNNF and compute the OR-index. We then enumerate compressed traces, and the valuations for each trace. The proof of Theorem 2.2 is the same except that we additionally use Proposition 4.1 before Proposition 4.3 to restrict to valuations of Hamming weight $\leq k$.

7 Applications

We now present two applications of our main results. Our first application recaptures the well-known enumeration results for MSO queries on trees [6, 23]. The second application describes links to factorized databases and strengthens the enumeration result of [29].

MSO enumeration. Recall that the class of *monadic second-order* formulae (MSO) consists of first-order logical formulae extended with quantification over sets, see e.g. [27]. The *enumeration problem* for a fixed MSO formula $\phi(X_1, \ldots, X_k)$ with free second-order variables, given a structure I, is to enumerate the *answers* of ϕ on I, i.e., the k-tuples $(B_1, \ldots, B_k)$ of subsets of the domain of I such that I satisfies $\phi(B_1, \ldots, B_k)$. We measure the *data complexity* of this task, i.e., its complexity in the input structure, with the query being fixed.

It was shown by Bagan [6] that MSO query enumeration on *trees* and *bounded treewidth structures* can be performed with linear-time preprocessing and delay linear in each MSO assignment; in particular, if the free variables of the formula are first-order, then the delay is constant. This latter result was later re-proven by Kazana and Segoufin [25]. We show how to recapture this theorem from our main results. From the results of Courcelle and standard techniques (see, e.g., [23], Theorem 6.3.1 and Section 6.3.2), we restrict to binary trees.

▶ **Definition 7.1.** Let Γ be a finite alphabet. A Γ-*tree* T is a rooted unordered binary tree where each node $n \in T$ carries a label in Γ. We abuse notation and identify T to its node set. *MSO formulae on* Γ-*trees* are written on the signature consisting of one binary predicate for the edge relation and unary predicates for each label of Γ.

Let $\phi(X_1, \ldots, X_k)$ be an MSO formula on Γ-trees, and let T be a Γ-tree. We will show our enumeration result by building a structured circuit capturing the *assignments* of ϕ on T:

▶ **Definition 7.2.** A *singleton* on $X_1, \ldots, X_k$ and T is an expression of the form $\langle X_i : n \rangle$ with $n \in T$. An *assignment* on $X_1, \ldots, X_k$ and T is a set S of singletons: it defines a k-tuple $(B_1^S, \ldots, B_k^S)$ of subsets of T by setting $B_i^S := \{n \in T \mid \langle X_i : n \rangle \in S\}$ for each i. The *assignments* of ϕ on T are the assignments S such that T satisfies $\phi(B_1^S, \ldots, B_k^S)$.

We will enumerate assignments instead of answers: this makes no difference because we can always rewrite each assignment in linear time to the corresponding answer. We now state the key result: we can efficiently build circuits (with singletons as variable gates) that capture the assignments to MSO queries. (While these circuits are not augmented circuits, they are decomposable, so the definition of zero-suppressed semantics clearly extends.)

▶ **Theorem 7.3.** *For any fixed MSO formula $\phi(X_1, \ldots, X_k)$ on Γ-trees, given a Γ-tree T, we can build in time $O(|T|)$ a monotone d-DNNF circuit C in zero-suppressed semantics whose set $S(C)$ of assignments (as in Definition 3.6) is exactly the set of assignments of ϕ on T.*

Proof Sketch. We simplify ϕ to have a single free variable and limit to assignments on leaves as in [6], and rewrite ϕ to a deterministic tree automaton A using the result of Thatcher and Wright [35], in time independent of T (though the runtime is generally nonelementary in ϕ).

We then compute our circuit as a variant of the *provenance circuits* in our earlier work [4], observing that it is a d-DNNF thanks to the determinism of the automaton as in [5]. This second step is in $O(|A| \cdot |T|)$, so linear in T. A self-contained proof is given in [3]. ◀

Note that the resulting circuit is already in zero-suppressed semantics, and has no range gates. By continuing as in the proof of Theorem 2.1 (for free second-order variables) or of Theorem 2.2 (for free first-order variables), we deduce the MSO enumeration results of [6, 25]. Note that, once we have computed the tree automaton for the query and the circuit representation, our proof of the enumeration result is completely query-agnostic: we simply apply our enumeration construction on the circuit. Our proof also does not depend on the factorization forest decomposition theorem of [15] used by [25]; it consists only of the simple circuit manipulation and indexing that we presented in Sections 4–6. Note that the delay is in $O(k \cdot |T|)$, with no large hidden constants, and $O(k)$ for first-order variables.

A limitation of our approach is that our memory usage bound includes a logarithmic factor in T, whereas [6, 25] show constant-memory enumeration. However, we can show that the circuit computed in Theorem 7.3 satisfies an *upwards-determinism* condition that allow us to replace the indexing scheme of Theorem 5.4 (our memory bottleneck) by a more efficient index. We can thus reprove the constant-memory enumeration of [6, 25]: see [3].

Factorized representations. Our second application is the *factorized representations* of [29], a concise way to represent database relations [1] by "factoring out" common parts. The atomic factorized relations are the empty relation $\emptyset$, the relation $\langle\rangle$ containing only the empty tuple, and singletons $\langle A : a\rangle$ where A is an attribute and a is an element. Larger relations are built using the relational union and Cartesian product operators on sub-relations with compatible schemas. For example, $\langle A_1 : a_1\rangle \times (\langle A_2 : a_2\rangle \cup \langle A_2 : a_2'\rangle)$ is a factorized representation of the relation on attributes A_1, A_2 containing the tuples (a_1, a_2) and (a_1, a_2'). A *d-representation* is a factorized representation given as a DAG, to reuse common sub-expressions. We show that d-representations can be seen as circuits in zero-suppressed semantics:

▶ **Lemma 7.4.** *For any d-representation D, let C be the monotone circuit obtained by replacing $\times$ and $\cup$ by AND and OR, replacing $\emptyset$ and $\langle\rangle$ by AND-gates and OR-gates with no inputs, and keeping singletons as variables. Then all AND-gates of C are decomposable, and $S(C)$ (defined as in Section 3) is exactly the database relation represented by D.*

Hence, our results in Theorem 2.2 can be rephrased in terms of factorized representations:

▶ **Theorem 7.5.** *The tuples of a deterministic d-representation D over a schema $\mathcal{S}$ can be enumerated with linear-time preprocessing, delay $O(|\mathcal{S}|)$, and memory $O(|\mathcal{S}| \log |D|)$.*

Note that the existing enumeration result on factorized representations (Theorem 4.11 of [29]) achieves a constant memory bound, unlike ours, but it applies only to deterministic d-representations that are *normal* (Definition 4.6 of [29]), which we do not assume. Normal d-representations are intuitively pruned and collapsed circuits where *no OR-gate is an input to an OR-gate*: this assumption avoids the need, e.g., for the constructions of Section 5.

8 Conclusion

We have shown how to enumerate satisfying valuations for the structured d-DNNF circuits used in AI, with linear preprocessing and delay linear in each valuation (so constant delay for constant Hamming weight). We applied this to factorized databases, and to MSO query

enumeration [6, 23]. Beyond this, however, our method implies efficient enumeration for all knowledge compilation problems that compile to structured d-DNNFs (see Introduction).

A natural question is to extend our constructions for other tasks, e.g., computing the i-th valuation [6, 9]; managing updates [28]; or enumerating in order of weight, or in lexicographic order: this latter problem is open for MSO [33, Section 6.1] but results are known for factorized representations following an f-tree [10]. Another direction is to strengthen our results to constant-memory enumeration on all d-DNNFs, or generalize them to other classes. We also plan to study practical implementations, because our construction only performs simple and modular transformations on input circuits, with no hidden large constants.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. Addison-Wesley, 1995. URL: http://webdam.inria.fr/Alice/pdfs/all.pdf.

2 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

3 Antoine Amarilli, Pierre Bourhis, Louis Jachiet, and Stefan Mengel. A circuit-based approach to efficient enumeration, 2017. URL: https://arxiv.org/abs/1702.05589, arXiv:1702.05589.

4 Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Provenance circuits for trees and treelike instances. In *ICALP*, 2015. URL: https://arxiv.org/abs/1511.08723.

5 Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Tractable lineages on treelike instances: Limits and extensions. In *PODS*, 2016. URL: https://arxiv.org/abs/1604.02761.

6 Guillaume Bagan. MSO queries on tree decomposable structures are computable with linear delay. In *CSL*, 2006.

7 Guillaume Bagan, Arnaud Durand, Emmanuel Filiot, and Olivier Gauwin. Efficient enumeration for conjunctive queries over X-underbar structures. In *CSL*, 2010. URL: https://hal.inria.fr/hal-00489955.

8 Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *CSL*, 2007. URL: http://www.logique.jussieu.fr/~durand/webperso/papers/BDGlongversion.pdf.

9 Guillaume Bagan, Arnaud Durand, Etienne Grandjean, and Frédéric Olive. Computing the jth solution of a first-order query. *ITA*, 42(1), 2008. URL: https://hal-univ-diderot.archives-ouvertes.fr/file/index/docid/221730/filename/bdgo.pdf.

10 Nurzhan Bakibayev, Tomáš Kočiskỳ, Dan Olteanu, and Jakub Závodnỳ. Aggregation and ordering in factorised databases. *PVLDB*, 2013. URL: https://www.cs.ox.ac.uk/dan.olteanu/papers/bkoz-vldb13-with-response.pdf.

11 David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *JCSS*, 41(3), 1990. URL: http://www.sciencedirect.com/science/article/pii/002200009090022D.

12 Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. On compiling CNFs into structured deterministic DNNFs. In *SAT*, 2015. URL: http://www.dcs.bbk.ac.uk/~florent/publi/cnf-to-ddnnf-upper-bound.pdf.

13 Simone Bova and Stefan Szeider. Circuit treewidth, sentential decision, and query compilation. In *PODS*, 2017. URL: https://arxiv.org/abs/1701.04626.

14 Arthur Choi and Adnan Darwiche. Dynamic minimization of sentential decision diagrams. In *AAAI*, 2013. URL: http://reasoning.cs.ucla.edu/fetch.php?id=128&type=pdf.

15 Thomas Colcombet. A combinatorial theorem for trees. In *ICALP*, 2007. URL: https://www.irif.fr/~colcombe/Publications/ICALP07-colcombet.pdf.

16 Bruno Courcelle. Linear delay enumeration and monadic second-order logic. *Discrete Applied Mathematics*, 157(12), 2009. URL: https://www.labri.fr/perso/courcell/Textes/LinDelayEnum.pdf.

17 Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Applied Non-Classical Logics*, 11(1-2), 2001. doi:10.3166/jancl.11.11-34.

18 Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 17, 2002. URL: https://www.jair.org/media/989/live-989-2063-jair.pdf.

19 Arnaud Durand and Etienne Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *TOCL*, 8(4), 2007. URL: https://arxiv.org/abs/cs/0507020.

20 Arnaud Durand, Nicole Schweikardt, and Luc Segoufin. Enumerating answers to first-order queries over databases of low degree. In *PODS*, 2014. URL: https://hal.inria.fr/hal-01070898/en.

21 Jinbo Huang and Adnan Darwiche. DPLL with a trace: From SAT to knowledge compilation. In *IJCAI*, 2005. URL: https://ijcai.org/Proceedings/05/Papers/0876.pdf.

22 Abhay Kumar Jha and Dan Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. *TCS*, 52(3), 2013.

23 Wojciech Kazana. *Query evaluation with constant delay.* PhD thesis, École normale supérieure de Cachan, 2013. URL: https://tel.archives-ouvertes.fr/tel-00919786/document.

24 Wojciech Kazana and Luc Segoufin. Enumeration of first-order queries on classes of structures with bounded expansion. In *PODS*. ACM, 2013. URL: https://hal.inria.fr/hal-00908779/en.

25 Wojciech Kazana and Luc Segoufin. Enumeration of monadic second-order queries on trees. *TOCL*, 14(4), 2013. URL: https://hal.archives-ouvertes.fr/docs/00/90/70/85/PDF/cdlin-survey.pdf.

26 Donald E. Knuth. *The Art of Computer Programming. Volume 4A: Combinatorial Algorithms, Part 1.* Addison-Wesley, 2005.

27 Leonid Libkin. *Elements of Finite Model Theory.* Springer, 2004.

28 Katja Losemann and Wim Martens. MSO queries on trees: enumerating answers under updates. In *CSL-LICS*, 2014. URL: http://www.theoinf.uni-bayreuth.de/download/lics14-preprint.pdf.

29 Dan Olteanu and Jakub Závodnỳ. Size bounds for factorised representations of query results. *TODS*, 40(1), 2015. URL: http://www.cs.ox.ac.uk/dan.olteanu/papers/oz-tods15.pdf.

30 Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In *IJCAI*, 2015. URL: http://reasoning.cs.ucla.edu/fetch.php?id=157&type=pdf.

31 Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, 2008. URL: http://aaai.org/Papers/AAAI/2008/AAAI08-082.pdf.

32 Ran Raz, Amir Shpilka, and Amir Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. *SIAM J. Comput.*, 38(4), 2008. doi:10.1137/070707932.

33 Luc Segoufin. A glimpse on constant delay enumeration (invited talk). In *STACS*, 2014. doi:10.4230/LIPIcs.STACS.2014.13.

34 Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264, 1973. URL: https://eudml.org/doc/151394.

35 James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Math. Systems Theory*, 2(1), 1968.

36 Kunihiro Wasa. Enumeration of enumeration algorithms. *CoRR*, abs/1605.05102, 2016. URL: https://arxiv.org/abs/1605.05102.
37 Ingo Wegener. *Branching programs and binary decision diagrams*. SIAM, 2000.

Automata-Based Stream Processing

Rajeev Alur[1], Konstantinos Mamouras[2], and Caleb Stanford[3]

1 **Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA**
`alur@cis.upenn.edu`
2 **Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA**
`mamouras@cis.upenn.edu`
3 **Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA**
`castan@cis.upenn.edu`

Abstract

We propose an automata-theoretic framework for modularly expressing computations on streams of data. With weighted automata as a starting point, we identify three key features that are useful for an automaton model for stream processing: expressing the regular decomposition of streams whose data items are elements of a complex type (e.g., tuple of values), allowing the hierarchical nesting of several different kinds of aggregations, and specifying modularly the parallel execution and combination of various subcomputations. The combination of these features leads to subtle efficiency considerations that concern the interaction between nondeterminism, hierarchical nesting, and parallelism. We identify a syntactic restriction where the nondeterminism is unambiguous and parallel subcomputations synchronize their outputs. For automata satisfying these restrictions, we show that there is a space- and time-efficient streaming evaluation algorithm. We also prove that when these restrictions are relaxed, the evaluation problem becomes inherently computationally expensive.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases weighted automata, Quantitative Regular Expressions, stream processing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.112

1 Introduction

Finite-state automata have been used very successfully to solve the problem of pattern matching in strings [1]. For simple patterns that are given as regular expressions, there have been proposed several pattern-matching algorithms based on Nondeterministic Finite Automata (NFAs) [31] or Deterministic Finite Automata (DFAs) [7] with strong efficiency guarantees. A particularly desirable feature of such automata-based algorithms is that they process the input text in one pass, i.e. by reading each letter of the input text consecutively from left to right, thus adhering to the so-called *streaming model* of computation [28].

Pattern-matching is one basic computational problem that arises in the context of data stream processing [14], i.e. the processing of data that arrives in real time at a high rate (e.g., for analyzing stock market data and web click-streams, or for monitoring sensor measurements and network traffic). To process data streams, the core computational problem that typically needs to be solved is the aggregation of parts of the stream into numerical values. For example, calculating the average price of a stock, monitoring the amount of network traffic an IP address has generated so far, or maintaining for a sensor the minimum and maximum


measurements it has recorded over the last 10 minutes. Given the usefulness of automata for finding patterns in streams of symbols, the question arises whether similar automata-based techniques can be employed for computing quantitative summaries of data streams.

We are thus led to consider weighted automata [19], which extend classical nondeterministic automata by annotating transitions with *weights* and can be used for the computation of simple quantitative properties on finite or infinite strings of symbols [10]. Weighted automata have found applications in speech and language processing [26], and they are also used for modeling systems and verifying quantitative properties of these systems [12]. However, the computational problems that are relevant for quantitative verification are analysis questions such as universality and equivalence. These questions are decidable only when the weights and the operations used on them are very simple [24, 2], so the studied models are usually equipped with a very limited set of primitive operations that are insufficient for expressing realistic streaming computations.

Since weighted automata are not expressive enough for typical streaming computations, our goal is to extend them for this purpose while maintaining the efficiency of their evaluation. First, we notice that the elements of data streams are typically not symbols from a finite alphabet but rather structured objects such as tuples of values. It is therefore necessary to work in the *symbolic* setting [33, 34]: the input elements belong to a potentially infinite alphabet D, and we consider a collection of primitive predicates on D for describing subclasses of elements using Boolean formulas over the primitive predicates. Additionally, realistic computations often involve the parsing of an input stream and aggregation of subcomputations; for example, we may want to subsample a sequence of sensor measurements by averaging them in groups of three consecutive measurements, and then compute the maximum measurement of every minute. Naturally describing such calculations requires that we allow *hierarchical nesting* of operations. In general, the required subcomputations may be disjoint from one another, and need to be executed in parallel. For example, suppose the automaton $\mathcal{A}_1$ describes a long-term average (e.g., over the last month) of a sensor measurement, $\mathcal{A}_2$ calculates a short-term average (e.g., over the last minute), and op is the "absolute difference" binary operation. Then, the construct $\mathrm{op}(\mathcal{A}_1, \mathcal{A}_2)$ describes the *parallel execution* of $\mathcal{A}_1$ and $\mathcal{A}_2$ and the *combination* of their results using the op operation. Thus, the overall computation outputs the distance between the short-term and long-term average. This construct for parallelism facilitates the modular description of computations.

Our contribution. Putting these desired features together in a model that supports nondeterministic parsing, hierarchical nesting of quantitative operations and modular parallelism is challenging. The core computational problem is the incremental evaluation of automata on unbounded data streams, and the goal is to provide an algorithm with strong space- and time-efficiency guarantees. We will establish formally that the naive combination of the desired features makes efficient evaluation impossible. Moreover, we will show that by restricting to unambiguous nondeterminism [9] and by constraining the parallel execution of $\mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ so that the automata $\mathcal{A}_i$ synchronize their outputs, we can achieve very efficient evaluation. More specifically, our main results are the following:

1. The evaluation problem for automata that allow ambiguous nondeterminism and nesting of quantitative operations requires space that is linear in the size of the input stream.
2. The evaluation problem for automata with unambiguous nondeterminism and unsynchronized parallel execution requires space that is exponential in the size of the automaton.
3. For automata that are unambiguous and allow only synchronized parallel execution, the evaluation problem requires space and time-per-element that is quadratic in the size of the automaton and independent of the size of the stream.

Related work. The features of our Streaming Automata (SAs) were inspired by the Quantitative Regular Expressions (QREs) of [5], which have constructs for parallelism and nesting of sequential aggregators. QREs were extended in [25] with streaming relational operations [22], and an efficient implementation was given for processing realistic workloads (Yahoo streaming benchmark [13] and NEXMark benchmark [32]). However, the evaluation algorithm of [5] and the implementation of [25] were not based on automata-theoretic techniques. A simplified version of the QREs of [5] without parameters allows a straightforward translation into our SAs that is very similar to the translation of unambiguous regexes into unambiguous NFAs. This translation is desirable not only because it gives rise to a cleaner evaluation algorithm, but also because it opens the door for systematic query optimization using automata-theoretic techniques, which could be explored in future research.

The model of Cost Register Automata (CRAs) was proposed in [4] and was shown in [5] to be expressively equivalent to QREs. However, CRAs cannot be used for the efficient evaluation of QREs, because the translation of QREs into CRAs incurs a doubly exponential blowup. The model of Streaming Automata that is proposed here is an appropriate setting for the efficient evaluation of QREs.

A two-level variant of weighted automata for infinite strings has recently been proposed [11] that can express long-run quantitative properties of a stream, for example, the average response time of a system. By restricting both the nesting depth (to 1) and the allowed aggregation operations, the model of [11] is shown to have decidable emptiness and universality problems. With the goal of modeling realistic streaming computations, we focus on arbitrary nesting and a general set of operations. We are therefore concerned primarily with evaluation complexity rather than decidability of these problems.

Symbolic automata and transducers [33, 34, 15, 16] have been introduced for matching and transforming strings over large or infinite alphabets. Our work builds on symbolic automata but instead addresses the problem of quantitative aggregation.

There is also related work on data words and data/register automata and their associated logics [23, 29, 18, 8]. These models operate on words over an infinite alphabet, which is typically of the form $\Sigma \times \mathbb{N}$, where Σ is a finite set of tags. They allow the comparison of infinite values using only the equality predicate. In contrast, our SAs do not allow binary predicates on stream elements, but instead allow a rich set of operations on the values.

More broadly, there is a vast line of research on efficient algorithms for the streaming model of computation. See the survey [28] and some illustrative works [27, 20, 3, 17, 6] that have been influential. The algorithms studied in this line of research are designed for specific problems (for example, finding the number of distinct elements in a stream) and typically use approximation and randomization. Our considerations here are orthogonal, and complementary, to the literature on streaming algorithms. We study the hierarchical nesting of several different kinds of aggregations, and we study the computational resources that are needed for parsing the stream and combining all intermediate results.

2 Streaming Automata

Symbolic input. Figure 1 shows two symbolic weighted automata over different inputs. $\mathcal{M}_1$ implements MaxBlockSum: on an input stream of natural numbers separated into (possibly empty) blocks by the separator 0, it returns the maximum sum of a block. As we may view $\mathcal{M}_1$ as a weighted automaton over the semiring $(\mathbb{N} \cup \{-\infty\}, \max, +)$, it does not yet introduce anything new to our model except the symbolic input. All transitions use the formal variable x to denote the current input data item, a natural number; the syntax

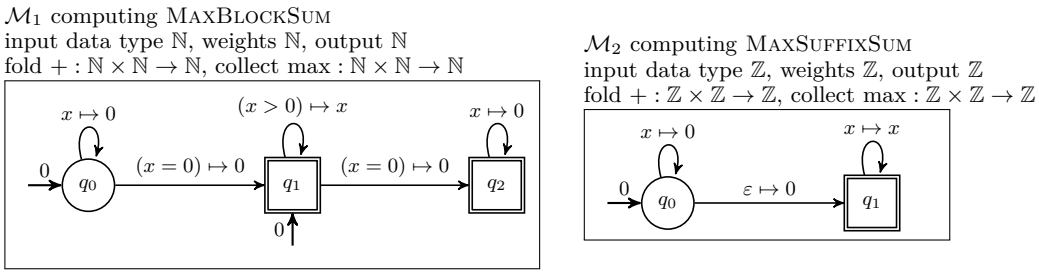


Figure 1 Weighted automata with symbolic input.

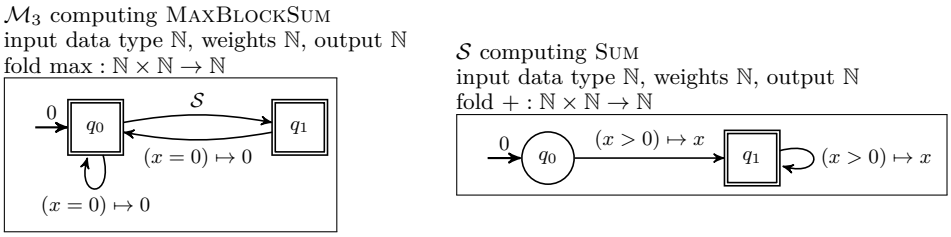


Figure 2 A streaming automaton employing hierarchy.

$\varphi(x) \mapsto \alpha(x)$ means that if x matches predicate φ, then the transition can be taken, and has weight $\alpha(x)$. We write simply $x \mapsto \alpha(x)$ if φ is True, i.e. if any x is allowed. A transition labeled with $\varepsilon \mapsto r$ matches the empty string and has weight r.

$\mathcal{M}_1$ starts at q_0 for some time, mapping each input x to weight 0 (effectively ignoring it). Then, it nondeterministically picks a block by transitioning to q_1 on input the separator $x = 0$. (q_1 is also a start state, which corresponds to the first block, before any 0 has occurred.) At q_1, all inputs matching the predicate $x > 0$ are assigned weight x. Finally, on input $x = 0$, the end of the block, it transitions to q_2, where future x are again assigned weight 0. $\mathcal{M}_1$ adds up (*folds*) all the assigned weights to obtain the total weight of the path, which is by construction the sum of the particular block chosen. The output of the automaton is the maximum weight (*collect*) over all paths.

$\mathcal{M}_2$ implements MaxSuffixSum: on an input stream of integers, it returns the maximum sum of a suffix of those integers. The input data type is now $\mathbb{Z}$ rather than $\mathbb{N}$. $\mathcal{M}_2$ (like $\mathcal{M}_1$) starts at q_0 and assigns inputs x to weight 0 for some time. Then, it nondeterministically guesses the start of the suffix by switching to q_1, where each future input x is assigned weight x. The fold operation is again $+$, so that the weight of the path is the sum of that particular suffix. The collect operation returns the max over all paths, i.e. over all suffixes.

Hierarchy. The nondeterminism of $\mathcal{M}_2$ is very natural: exactly where the best suffix starts cannot be known ahead of time, so we choose it nondeterministically. In contrast, since the input to $\mathcal{M}_1$ is *parsable* into a sequence of blocks, using nondeterminism to choose a block seems artificial. Instead, we would like to deterministically parse the stream into blocks, then call a subroutine (sum) on each block. Figure 2 shows how to do this in our model. First, the weighted automaton $\mathcal{S}$ is built to compute the sum of a nonempty input stream by straightforwardly folding with $+$. $\mathcal{M}_3$ parses the stream into blocks separated by 0 and calls $\mathcal{S}$ as a *subautomaton* on each block, where the weight of that transition is the return value of $\mathcal{S}$. All the block sums returned by $\mathcal{S}$ are now weights along a single path, and they are folded with the operation max.

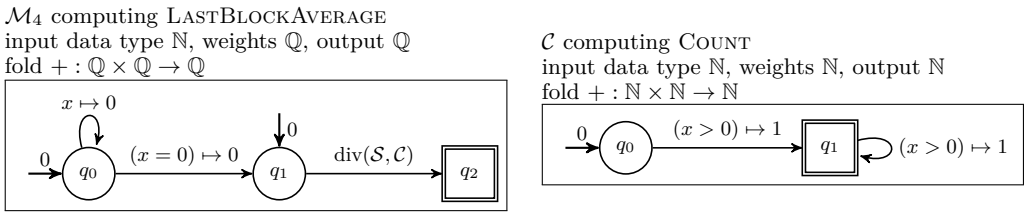


Figure 3 A streaming automaton employing parallelism.

The example of MaxBlockSum is a typical case where the two operations of a nondeterministic weighted automaton (fold $\otimes$ and collect $\oplus$) can be replaced by a hierarchy of two streaming automata, each of which is *unambiguous*: there is at most one accepting path on any given input string. The fold operation of $\mathcal{M}_1$ $(+)$ becomes the fold operation of $\mathcal{S}$, and the collect operation of $\mathcal{M}_1$ (max) becomes the fold operation of $\mathcal{M}_3$. Unambiguity implies that the collect operations in $\mathcal{M}_3$ and $\mathcal{S}$ are never used, and need not be specified.

Parallelism. After parsing a stream into blocks, multiple computations may be required on each block. For this purpose, in our model a transition may be labeled not just with a single subautomaton (as in $\mathcal{M}_3$), but with a call $\mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ where each $\mathcal{A}_i$ is a subautomaton. In a simple example, the stream is separated by 0 into blocks, and we want to report the average of the last block. Figure 3 gives an automaton $\mathcal{M}_4$ implementing this. On every 0 character $\mathcal{M}_4$ may nondeterministically guess that we are now going to the last block, and move from q_0 to q_1. It subsequently makes an invocation $\mathrm{div}(\mathcal{S}, \mathcal{C})$ to two subautomata. $\mathcal{S}$ (from Figure 2) returns the sum of the elements in the block if there is at least one, and $\mathcal{C}$ returns the count if there is at least one. $\mathrm{div} : \mathbb{N} \times \mathbb{N} \to \mathbb{Q}$ then divides the two results to get average. The *parallelism* arises because the stream is read into both $\mathcal{S}$ and $\mathcal{C}$ in parallel.

Like $\mathcal{M}_3$, $\mathcal{M}_4$ is unambiguous, with at most one accepting path on each input. $\mathcal{M}_4$ also satisfies *parallel-consistency*: in the call to $\mathrm{div}(\mathcal{S}, \mathcal{C})$, $\mathcal{S}$ and $\mathcal{C}$ were defined on the same input strings. Our definition of a *streaming automaton* requires both unambiguity and parallel-consistency; the necessity of these restrictions is justified by Section 4.

Formal definition

The general definition is parameterized by a *signature* $(\mathcal{D}, \mathcal{O}, D, \mathcal{P})$, where $\mathcal{D}$ is a collection of (possibly infinite) types, and $\mathcal{O}$ is a collection of operations $D_1 \times D_2 \times \cdots \times D_k \to D_{k+1}$ with each D_i a type in $\mathcal{D}$. We write $\mathcal{O}[D_1 \times D_2 \times \cdots \times D_k \to D_{k+1}]$ for the set of operations in $\mathcal{O}$ which are functions of the specific indicated function type. $D \in \mathcal{D}$ is a specific set for the input stream, and $\mathcal{P}$ is a set of predicates, which are identified with subsets of D. We require that $\mathcal{P}$ is closed under Boolean operations, and that satisfiability for $\varphi \in \mathcal{P}$ is decidable as in [34]. From this point, we assume the fixed signature $(\mathcal{D}, \mathcal{O}, D, \mathcal{P})$.

The class of *nondeterministic streaming automata* is defined hierarchically as $\mathrm{NSA} := \bigcup_{k=0}^{\infty} \mathrm{NSA}_k$. For $k \geq 0$, an element of NSA_k is a tuple $(Q, X, Y, \Delta, I, F, \otimes, \oplus)$, semantically representing a partial function from D^* to Y. Q is a finite set of *states*, $X \in \mathcal{D}$ is the *weight type*, $Y \in \mathcal{D}$ is the *output type*, and Δ is a set of *transitions*. Each transition goes from a state $q \in Q$ to a state $q' \in Q$, and has a *label*, which is one of three *kinds*: (i) A satisfiable predicate $\varphi \in \mathcal{P}$ and a *weight assignment* $\alpha \in \mathcal{O}[D \to X]$. (ii) An epsilon (ε) and a *weight* $x \in X$. (iii) A call to $\mathrm{op}(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m)$, where $\mathrm{op} \in \mathcal{O}[Y_1 \times Y_2 \times \cdots \times Y_m \to X]$ and each $\mathcal{A}_i \in \mathrm{NSA}_{k-1}$, such that the output type of $\mathcal{A}_i$ is Y_i. The weight of the transition in this case will be op applied to the outputs of the $\mathcal{A}_i$.

$I : Q \rightharpoonup Y$ is the *initialization function*, a partial function assigning an initial value to the computation. Its domain is the set of *initial states*, denoted $Q_I \subseteq Q$. Conversely, $F : Q \rightharpoonup X$ is the *final function*; it allows for slightly more flexibility than in our examples by appending a final weight to accepting paths. Its domain is the set of *final states* or *accepting states*, denoted $Q_F \subseteq Q$. The *fold operation* $\otimes \in \mathcal{O}[Y \times X \to Y]$ folds together the weights along a path, and the *collect operation* $\oplus \in \mathcal{O}[Y \times Y \to Y]$ combines the results of all accepting paths to arrive at a final output value. The operation $\oplus$ must be commutative and associative, and $\otimes$ must be left-distributive over $\oplus$.

The class NSA_0, in which there are no transitions of kind (iii), consists of symbolic weighted automata. A *subautomaton* of $\mathcal{A}$ is an automaton $\mathcal{A}_i \in \mathrm{NSA}_{k-1}$ appearing in a transition of kind (iii) in A. The *size* of A is the sum of the number of states $|Q|$, the number of transitions $|\Delta|$, and the sizes of all the subautomata, counted with multiplicity. Effectively, an automaton must be written down once for every time it is used.

As in the examples, the automaton $\mathcal{A}$ is semantically interpreted as a function $[\![\mathcal{A}]\!] : L(\mathcal{A}) \to Y$, where $L(\mathcal{A}) \subseteq D^*$ is the regular *language* of $\mathcal{A}$. $L(\mathcal{A})$ and $[\![\mathcal{A}]\!]$ are defined recursively by also defining $L(\tau)$ and $[\![\tau]\!]$ for each transition τ of the automaton. (i) For a transition τ labeled with predicate $\varphi \subseteq D$ and weight assignment $\alpha : D \to X$, $L(\tau) = \{d \in D \mid \varphi(d)\}$, and $[\![\tau]\!](d) = \alpha(d)$. (ii) For an epsilon transition τ with weight $x \in X$, $L(\tau) = \{\varepsilon\}$ and $[\![\tau]\!](\varepsilon) = x$. (iii) Finally, for a transition τ labeled with $\mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_m)$, the language $L(\tau) = L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_m)$, and for any string $s \in L(\tau)$, $[\![\tau]\!](s) = \mathrm{op}([\![\mathcal{A}_1]\!](s), \ldots, [\![\mathcal{A}_m]\!](s))$.

For an automaton $\mathcal{A} \in \mathrm{NSA}_k$, a *path* on input $s \in D^*$ consists of a sequence of states $q_0, q_1, q_2, \ldots, q_n \in Q$, a sequence of strings $s_1, s_2, \ldots, s_n \in D^*$, and a sequence of transitions $\tau_1, \tau_2, \ldots, \tau_n \in \Delta$, such that $q_0 \in Q_I$, $s = s_1 s_2 \ldots s_n$, and for each i, τ_i is a transition from q_{i-1} to q_i such that $s_i \in L(\tau_i)$. A path is *accepting* if $q_n \in Q_F$. The *language* $L(\mathcal{A})$ is the set of strings s for which there exists an accepting path on input s. The *weight* of an accepting path is, with left-to-right evaluation order, $I(q_0) \otimes [\![\tau_1]\!](s_1) \otimes [\![\tau_2]\!](s_2) \otimes \cdots \otimes [\![\tau_n]\!](s_n) \otimes F(q_n) \in Y$.

An *implicit ε-transition* is a transition τ with $\varepsilon \in L(\tau)$. $\mathcal{A}$ is *well-formed* if it has no implicit ε-transition cycles, and all of its subautomata are well-formed. Finally, the evaluation of $\mathcal{A}$ on input $s \in L(\mathcal{A})$ is given by $[\![\mathcal{A}]\!](s) := y_1 \oplus \cdots \oplus y_N \in Y$, where $y_1, \ldots, y_N$ are the weights of *all* (finitely many) distinct accepting paths on input s. As $\oplus$ is commutative and associative, this is well-defined.

Streaming automata. We recursively say that an NSA $\mathcal{A}$ is *unambiguous* if there is at most one accepting path on every input string, and each subautomaton of $\mathcal{A}$ is unambiguous. An NSA $\mathcal{A}$ is called *parallel-consistent* if, at every transition of kind (iii) labeled with $\mathrm{op}(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m)$, $L(\mathcal{A}_1) = L(\mathcal{A}_2) = \cdots = L(\mathcal{A}_m)$, and every subautomaton is parallel-consistent. A *streaming automaton (SA)* is an NSA $\mathcal{A}$ that is unambiguous and parallel-consistent. The collect operation $\oplus$ of an SA may be left off, as it is never invoked. We additionally assume that every SA is *trim*: every state has an accepting path which goes through it, and all subautomata are trim.

Checking if an NSA is an SA. Both of the two restrictions (unambiguity and parallel-consistency) can be checked efficiently. The main idea is to assign to each subautomaton $\mathcal{A}$ an *underlying NFA* $\mathrm{NFA}(\mathcal{A})$, such that $L(\mathcal{A}) = L(\mathrm{NFA}(\mathcal{A}))$, from the bottom up. Given an NSA $\mathcal{A}$, the algorithm recursively verifies that $\mathcal{A}$ is unambiguous and parallel-consistent, and also returns the NFA $\mathrm{NFA}(\mathcal{A})$ such that $L(\mathcal{A}) = L(\mathrm{NFA}(\mathcal{A}))$. Assume this has been done for all subautomata of $\mathcal{A}$. Checking parallel-consistency of a transition labeled $\mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ is then the equivalence problem for the unambiguous NFAs $\mathrm{NFA}(\mathcal{A}_1), \ldots, \mathrm{NFA}(\mathcal{A}_m)$; ex-

actly this problem is solved in polynomial time by a nontrivial algorithm of [30]. Once parallel-consistency is established, we form NFA($\mathcal{A}$) by replacing each transition labeled with op($\mathcal{A}_1, \ldots, \mathcal{A}_m$) with ε-transitions to and from a copy of NFA($\mathcal{A}_1$). Crucially, we assume parallel-consistency in only using $\mathcal{A}_1$. This guarantees that the NFA is linear in the size of $\mathcal{A}$, and avoids the alternative of constructing an NFA for $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_m)$. The construction preserves accepting paths, so $L(\mathcal{A}) = L(\text{NFA}(\mathcal{A}))$, and if one is unambiguous, both are. Finally, checking that NFA($\mathcal{A}$) is unambiguous is a reachability check in NFA($\mathcal{A}$) $\times$ NFA($\mathcal{A}$).

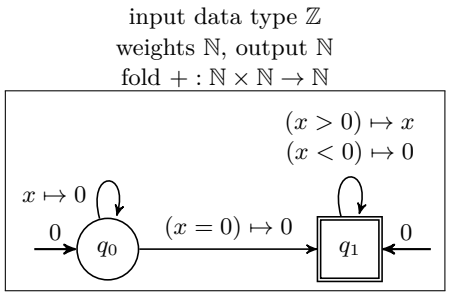
The necessary operations for the algorithm to work lift to the symbolic setting given the decidability restrictions on the predicates. See e.g. Corollary 1 of [34].

3 Evaluation Algorithm

In this section we present a space- and time-efficient evaluation algorithm for streaming automata, i.e. NSAs that are unambiguous and parallel-consistent. We will show that for such automata the space footprint of the evaluation algorithm and the time required to process each element are independent of the size of the stream and quadratic in the size of the automaton. As we will see in Section 4, both these syntactic restrictions on automata are necessary for the efficiency guarantees that we present.

Given an SA $\mathcal{A}$ and a sequence w of data items, the computation of $[\![\mathcal{A}]\!](w)$ amounts to discovering a global hierarchical path for w that may span several levels of subautomata and performing incrementally the aggregations that are prescribed by the top level and all subautomata. The crucial challenge is that the unambiguous nondeterminism of $\mathcal{A}$ requires the exploration of all possible paths *in parallel*. It is not obvious how this can be accomplished using a small amount of space, and indeed Theorem 5 in the next section shows that this is impossible in the presence of ambiguous nondeterminism. For plain NFAs or weighted automata, ambiguous nondeterminism is not an issue, because when two tokens end up at the same state during evaluation they can be merged. For streaming automata, however, such merging is not possible. The main insight is that unambiguity guarantees that no two tokens will ever end up at the same state, even at the lowest level of the automaton. As the evaluation algorithm explores each tentative path, it maintains a *stack of values* for that path, which holds the partial aggregates for the subpaths that have been discovered so far. We can think of these stacks as "execution tokens" that are updated whenever a simple transition occurs (upon consumption of a data item), and which are passed to subautomata as a way to implement the recursive definition of global accepting paths.

Before presenting the technical details, let us give a very high-level description of the evaluation algorithm and its correctness proof. First, we will introduce the notion of a *configuration*, which describes the assignment of stack tokens to the active states of the automaton. This is a generalization of configurations for NFAs, which only indicate the active states. We will define a semantics for configurations, which summarizes the accepting paths from active states as well as the computations that are performed along these paths. Then, the correctness proof of the algorithm can be reduced to establishing a simple semantic property for configurations: if C is the current configuration and C' is the configuration that the evaluation algorithm computes from C after consuming the data item d, then $[\![C']\!](w) = [\![C]\!](dw)$ for every possible suffix w. The presence of several nested levels of subautomata presents a major challenge for proving this property, since a subautomaton potentially has to compute simultaneously on several subsequences of the stream seen so far (we call these subsequences "parallel input threads").



start/next	configuration	input threads
[9]	$q_0 : [9, 0]$ $q_1 : [9, 0]$	{[9] ε}
2	$q_0 : [9, 0]$ $q_1 : [9, 2]$	{[9] 2}
-1	$q_0 : [9, 0]$ $q_1 : [9, 2]$	{[9] 2 -1}
3	$q_0 : [9, 0]$ $q_1 : [9, 5]$	{[9] 2 -1 3}
0	$q_0 : [9, 0]$ $q_1 : [9, 0]$	{[9] 2 -1 3 0}
6	$q_0 : [9, 0]$ $q_1 : [9, 6]$	{[9] 2 -1 3 0 6}

Figure 4 Example evaluation of an SA on one input thread.

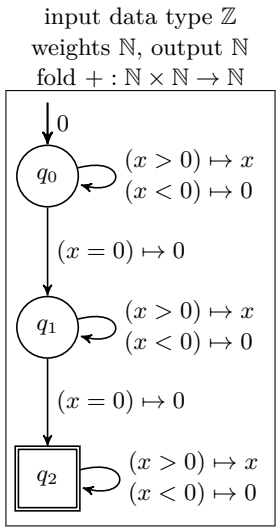
▶ **Example 1.** The automaton of Figure 4 computes on a stream of integers and outputs the sum of all strictly positive numbers that have occurred after the last occurrence of a 0 (or from the start if no 0 has occurred yet). We start the execution by supplying a *context stack*, which holds the partial aggregations of upper levels (if there are any), and then we supply the sequence of data items. The context stack [9] of this example is initialized by pushing the aggregate 0 onto it, and then every time an element is consumed the aggregate at the top of the stack is appropriately updated.

▶ **Example 2.** The automaton of Figure 5 computes on a stream of integers and outputs the sum of all strictly positive numbers that have occurred as long as there are exactly two occurrences of a 0. We can compute on several parallel input threads by supplying a new context stack every time we want to spawn a new thread of execution. Figure 5 shows an example execution with three different input threads. By starting a new input thread after the occurrence of a 0 we guarantee that there is at most one stack token on each state.

Epsilon transitions can be eliminated in a bottom-up fashion with a variant of the standard ε-elimination construction for weighted automata [19]. We consider in this section automata that are free of both explicit and implicit ε-transitions, and we assume w.l.o.g. that every invocation $\mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ has its own call state p, from which no other transition emanates.

Suppose that V is the type of all *values*. Let St be the type of all finite stacks of values, and [] be the empty stack. We consider the total operation $\mathsf{push} : \mathsf{St} \times V \to \mathsf{St}$, and the partial operations $\mathsf{pop} : \mathsf{St} \rightharpoonup \mathsf{St}$ and $\mathsf{top} : \mathsf{St} \rightharpoonup V$. The operations pop and top are undefined on the empty stack. We write $s.\mathsf{push}(x)$ to denote the application of push on the stack s and the value x. Similarly, we write $s.\mathsf{pop}$ and $s.\mathsf{top}$ for the other operations. For example, we have $[].\mathsf{push}(x).\mathsf{push}(y) = [x].\mathsf{push}(y) = [x, y]$ and $[x, y].\mathsf{pop}.\mathsf{top} = [x].\mathsf{top} = x$. We write $\mathsf{St}[X_1, \ldots, X_{n-1}, X_n]$ for the type of stacks of size n whose top element is of type X_n, the next-to-top element is of type X_{n-1} and so on. We call all types of this form *bounded stack types*. If $T = \mathsf{St}[X_1, \ldots, X_n]$ then we write $T@[X_{n+1}, \ldots, X_{n+m}]$ to denote the type $\mathsf{St}[X_1, \ldots, X_n, X_{n+1}, \ldots, X_{n+m}]$. We also abbreviate $T@[X_{n+1}]$ by $T@X_{n+1}$.

The *rank* of an SA $\mathcal{A}$ is the smallest k such that $\mathcal{A} \in \mathrm{NSA}_k$, or in other words the nesting depth of the automaton. We define the notion of a configuration for an automaton by induction on its rank. For an automaton $\mathcal{A} = (Q, X, Y, \Delta, I, F, \otimes)$ of rank 0 and a bounded stack type T, an $(\mathcal{A}, T)$-*configuration* is a partial map $C : Q \rightharpoonup T@Y$; we denote the domain $\mathrm{dom}(C)$. Intuitively, the configuration describes the placement of stack tokens on some of the states of $\mathcal{A}$. For an automaton $\mathcal{A} = (Q, X, Y, \Delta, I, F, \otimes)$ of rank strictly greater than 0, an $(\mathcal{A}, T)$-configuration C is a vector consisting of a partial function $C_0 : Q \rightharpoonup T@Y$ and a



start/next	configuration	input threads
[90]	$q_0 : [90, 0]$	{[90] ε}
3	$q_0 : [90, 3]$	{[90] 3}
−2	$q_0 : [90, 3]$	{[90] 3 −2}
0	$q_1 : [90, 3]$	{[90] 3 −2 0}
[70]	$q_0 : [70, 0]$	{[90] 3 −2 0,
	$q_1 : [90, 3]$	[70] ε}
2	$q_0 : [70, 2]$	{[90] 3 −2 0 2,
	$q_1 : [90, 5]$	[70] 2}
−1	$q_0 : [70, 2]$	{[90] 3 −2 0 2 −1,
	$q_1 : [90, 5]$	[70] 2 −1}
6	$q_0 : [70, 8]$	{[90] 3 −2 0 2 −1 6,
	$q_1 : [90, 11]$	[70] 2 −1 6}
0	$q_1 : [70, 8]$	{[90] 3 −2 0 2 −1 6 0,
	$q_2 : [90, 11]$	[70] 2 −1 6 0}
[30]	$q_0 : [30, 0]$	{[90] 3 −2 0 2 −1 6 0,
	$q_1 : [70, 8]$	[70] 2 −1 6 0,
	$q_2 : [90, 11]$	[30] ε}
−4	$q_0 : [30, 0]$	{[90] 3 −2 0 2 −1 6 0 −4,
	$q_1 : [70, 8]$	[70] 2 −1 6 0 −4,
	$q_2 : [90, 11]$	[30] −4}
−5	$q_0 : [30, 0]$	{[90] 3 −2 0 2 −1 6 0 −4 −5,
	$q_1 : [70, 8]$	[70] 2 −1 6 0 −4 −5,
	$q_2 : [90, 11]$	[30] −4 −5}
4	$q_0 : [30, 4]$	{[90] 3 −2 0 2 −1 6 0 −4 −5 4,
	$q_1 : [70, 12]$	[70] 2 −1 6 0 −4 −5 4,
	$q_2 : [90, 15]$	[30] −4 −5 4}

Figure 5 Example evaluation of an SA on several input threads.

subconfiguration for every subautomaton occurrence. More specifically, for every transition $(p, \mathrm{op}_i(\mathcal{A}_{i1}, \mathcal{A}_{i2}, \ldots, \mathcal{A}_{in}), q)$ of $\mathcal{A}$, the configuration C specifies an $(\mathcal{A}_{i1}, T@Y)$-configuration C_{i1} and a $(\mathcal{A}_{ij}, \mathsf{St}[])$-configuration C_{ij} for $j = 2, \ldots, n$. That is, the configuration describes the placement of stack tokens on the top-level states and specifies subconfigurations for the subautomata occurrences. We write $\mathrm{Cfg}\langle \mathcal{A}, T\rangle$ for the set of all $(\mathcal{A}, T)$-configurations.

For an automaton $\mathcal{A} = (Q, X, Y, \Delta, I, F, \otimes)$ and an $(\mathcal{A}, T)$-configuration C, we will define simultaneously C-paths, unambiguity of C, and the denotation $[\![C]\!] : D^* \rightharpoonup T@Y$ by induction on the rank of $\mathcal{A}$. A C-path is a path starting from the configuration C.

- ***Automaton $\mathcal{A}$ of rank 0***: A *C-path* (labeled with $d_1 d_2 \ldots d_n \in D^*$) is a sequence of the following form: $q_0 \to^{\phi_1/\sigma_1}_{d_1/x_1} q_1 \to^{\phi_2/\sigma_2}_{d_2/x_2} \cdots \to^{\phi_n/\sigma_n}_{d_n/x_n} q_n$, such that $q_0 \in \mathrm{dom}(C)$ and $(q_{i-1}, \phi_i, \sigma_i, q_i) \in \Delta$ with $\phi_i(d_i) = \mathit{true}$ and $x_i = \sigma_i(d_i)$ for every $i = 1, \ldots, n$. A C-path is said to be *accepting* if it ends with an accepting state. The *weight* of an accepting C-path is defined to be the value $\mathrm{fold}(y, \otimes, x_1 x_2 \ldots x_n x_{n+1})$ where $y = C(q_0).\mathsf{top}$ and $x_{n+1} = F(q_n)$. The configuration C is *unambiguous* if for every label $w \in D^*$ there is at most one accepting C-path labeled with w. For an unambiguous configuration C, the denotation $[\![C]\!] : D^* \rightharpoonup T@Y$ is defined as follows: if there is an accepting C-path π labeled with w starting with the state q, then $[\![C]\!]\ w = s.\mathsf{pop}.\mathsf{push}(y)$ where $s = C(q)$ is the initial stack and y is the weight of π.
- ***Automaton $\mathcal{A}$ of rank greater than 0***: A *top-level C-path* is a sequence of top-level transitions that can be of the following two forms:

$$p \to^{\phi/\sigma}_{d/x} q \qquad \text{where } (p, \phi, \sigma, q) \in \Delta \text{ with } \phi(d) = \mathit{true} \text{ and } x = \sigma(d)$$

$$p \to^{\mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_n)}_{w/x} q \qquad \text{where } w \neq \varepsilon \text{ and } (p, \mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_n), q) \in \Delta \text{ with } x = \mathrm{op}([\![\mathcal{A}_1]\!]\ w, \ldots, [\![\mathcal{A}_n]\!]\ w)$$

that starts with a state in the domain of C_0. Now, a *cross-level C-path* is a sequence of

top-level transitions with an additional prefix called a cross-level transition:

$$\rightarrow_{w/t}^{\mathrm{op}(\mathcal{A}_{i1},\ldots,\mathcal{A}_{in})} q \quad \text{where } w \neq \varepsilon \text{ and } (p, \mathrm{op}(\mathcal{A}_{i1},\ldots,\mathcal{A}_{in}), q) \in \Delta \text{ for some state } p$$
$$s_1 = [\![C_{i1}]\!](w) : T@[Y, Z_1] \text{ and } s_j = [\![C_{ij}]\!](w) : [Z_j] \text{ for } j = 2,\ldots,n$$
$$t = s_1.\mathsf{pop}.\mathsf{pop}.\mathsf{push}(s_1.\mathsf{pop}.\mathsf{top} \otimes \mathrm{op}(z_1,\ldots,z_n)) \text{ where } z_j = s_j.\mathsf{top}$$

Such a prefix summarizes a path in the lower levels, and its annotation w/t specifies both a label $w \neq \varepsilon$ and a stack $t : T@Y$ for continuing at the top level. The label of a path is the concatenation from left to right of the strings over D that annotate the transitions. The *weight* of a top-level C-path is defined as in the 0-rank case, and the *weight* of a cross-level C-path is similar but the initial stack is specified by the first (cross-level) transition. The configuration C is *unambiguous* if it satisfies the following two conditions:

1. For every label $w \in D^*$ there is at most one accepting C-path (top-level or cross-level) labeled with w.
2. For every transition $(p, \mathrm{op}(\mathcal{A}_{i1},\ldots,\mathcal{A}_{in}), q)$, the denotations $[\![C_{i1}]\!]$, …, $[\![C_{in}]\!]$ have equal domains.

For an unambiguous configuration C, the denotation $[\![C]\!] : D^* \rightharpoonup T@Y$ is defined as follows: if there is an accepting C-path π (top-level or cross-level) labeled with w, then $[\![C]\!]\ w = s.\mathsf{pop}.\mathsf{push}(y)$ where s is the initial stack (specified by C_0 for top-level C-paths, and by the initial transition for the cross-level C-paths) and y is the weight of π.

For an SA $\mathcal{A} = (Q, X, Y, \Delta, I, F, \otimes)$ and a bounded stack type T, we define the denotation $\langle\!\langle \mathcal{A} \rangle\!\rangle_T : T \to (D^* \rightharpoonup T@Y)$ as $\langle\!\langle \mathcal{A} \rangle\!\rangle_T\ s\ w = s.\mathsf{push}([\![\mathcal{A}]\!]\ w)$.

Figure 6 describes the evaluation algorithm for the base case of a streaming automaton of rank 0. Observe that the algorithm specifies a procedure $\mathbf{next}(d)$ for consuming the element d, and a procedure $\mathbf{start}(s)$ for starting a new input thread given the context stack s. This generalization of being able to start several parallel input threads is necessary when the automaton is nested beneath other upper-level automata.

Figure 7 describes the evaluation algorithm for the case of a streaming automaton of rank strictly greater than 0. The interface is the same as for the base case: there are procedures $\mathbf{start}(s)$ and $\mathbf{next}(d)$. The main difference is that the algorithm in this case has to deal with the invocation transitions: every time a token is at a call state the corresponding subautomata are restarted, and every time the subautomata have output the corresponding return state is updated with the output stack.

▶ **Lemma 3.** Let $\mathcal{A} = (Q, X, Y, \Delta, I, F, \otimes)$ be an SA and T be a bounded stack type. Then:

1. Let C be an unambiguous $(\mathcal{A}, T)$-configuration and s a stack of type T so that $[\![C]\!]$ and $\langle\!\langle \mathcal{A} \rangle\!\rangle_T(s)$ are disjoint. Then, the configuration $\mathbf{start}(C, s)$, as described operationally in Figure 6 and Figure 7, is unambiguous and satisfies $[\![\mathbf{start}(C, s)]\!] = [\![C]\!] \sqcup \langle\!\langle \mathcal{A} \rangle\!\rangle_T(s)$.
   Notation: If f and g are partial functions with disjoint domains, the partial function $f \sqcup g$ has domain $\mathrm{dom}(f) \cup \mathrm{dom}(g)$ and agrees with both f and g.
2. Let C be an unambiguous $(\mathcal{A}, T)$-configuration and $d \in D$. Then, the configuration $\mathbf{next}(C, d)$, as described operationally in Figure 6 and Figure 7, is unambiguous and satisfies $[\![\mathbf{next}(C, d)]\!]\ w = [\![C]\!]\ dw$ for all sequences $w \in D^*$.

Lemma 3 establishes the main semantic property for configurations that is needed for proving the correctness of the evaluation algorithm.

▶ **Theorem 4.** *The streaming algorithm of Figure 6 and Figure 7 solves the evaluation problem for streaming automata. The space footprint of the algorithm and the processing time per element are independent of the length of the stream and quadratic in the size of the automaton (assuming that the data types require unit space and the operations unit time).*

Streaming automaton $\mathcal{A} = (Q, X, Y, \Delta, I, F, \otimes)$ of rank 0 & bounded stack type T.
state: unambiguous $(\mathcal{A}, T)$-configuration $C : \mathrm{Cfg}\langle\mathcal{A}, T\rangle$, that is, $C : Q \rightharpoonup T@Y$

initialize($\mathrm{Cfg}\langle\mathcal{A}, T\rangle$ *this*) :
  $\mathit{this}.C := \bot$ // empty configuration

$T@Y$ **output**($\mathrm{Cfg}\langle\mathcal{A}, T\rangle$ *this*) :
  foreach $q \in Q_F$ do // iterate over final states
    if ($\mathit{this}.C(q)$ is defined) then return $\mathit{this}.C(q)$
  return nil

start($\mathrm{Cfg}\langle\mathcal{A}, T\rangle$ *this*, $T\ s$) : // precondition: $[\![\mathit{this}.C]\!]$ and $\langle\!\langle\mathcal{A}\rangle\!\rangle_T(s)$ are disjoint
  foreach $q \in Q_I$ do // place token on each initial state
    // $\mathit{this}.C(q)$ must be undefined
    $\mathit{this}.C(q) := s.\mathsf{push}(I(q))$

next($\mathrm{Cfg}\langle\mathcal{A}, T\rangle$ *this*, $D\ d$) :
  $\mathrm{Map}\langle Q, T@Y\rangle\ C_{\mathrm{next}} := \bot$
  foreach transition (p, ϕ, σ, q) in Δ do
    if $\phi(d) = \mathit{true}$ then
      $T@Y\ s := \mathit{this}.C(p)$ // current stack
      $Y\ y := s.\mathsf{top} \otimes \sigma(d)$ // new value
      $C_{\mathrm{next}}(q) := s.\mathsf{pop}.\mathsf{push}(y)$ // new stack
  $\mathit{this}.C := C_{\mathrm{next}}$

Figure 6 General evaluation algorithm for an SA of rank 0.

The guarantees of Theorem 4 apply unconditionally to the case of constant-size types and operations (e.g., integers and floating-point numbers specified by machine architectures). In the case of infinite data types, one may need to account for the additional complexity of computing on their unbounded values to obtain a more precise analysis. In any case, however, Theorem 4 can be understood as saying that the computational overhead of parsing the input stream and combining the intermediate results is not significant.

4 Lower Bounds

The efficient evaluation algorithm of the previous section depends crucially on the unambiguity and parallel-consistency of the automata. In fact, both these syntactic restrictions are essential for efficient evaluation. More specifically, ambiguous nondeterminism can make the streaming space complexity of evaluation linear in the size of stream. Moreover, the absence of parallel-consistency allows the encoding of unambiguous regular expressions with intersection. The streaming matching problem for such expressions requires space that is exponential in the size of the expression. These lower bounds highlight the difficulty of efficiently evaluating quantitative automata that allow for the interaction between nondeterminism and parallelism.

Consider a stream of natural numbers and the problem MinAvgSuffix for the streaming computation of the function $f(x_1x_2\ldots x_n) = \min_{i=1}^{n} \mathrm{average}(x_i, x_{i+1}, \ldots, x_n)$, where $x_1x_2\ldots x_n$ is the stream seen so far. An NSA similar to M_2 of Figure 1 may be constructed which computes from each suffix a pair $(\mathrm{sum}, \mathrm{count})$, and that is nested inside an automaton dividing the components of the pair to obtain the average. Since this automaton computes MinAvgSuffix, the following theorem asserts a lower bound for the evaluation problem of NSAs with two-level nesting but without parallelism.

Streaming automaton $\mathcal{A} = (Q, X, Y, \Delta, I, F, \otimes)$ **of rank** > 0 **& bounded stack type** T**.**
state: unambiguous $(\mathcal{A}, T)$-configuration $C : \mathrm{Cfg}\langle \mathcal{A}, T\rangle$

initialize($\mathrm{Cfg}\langle \mathcal{A}, T\rangle$ *this*) :
  $\mathit{this}.C_0 := \bot$ // no top-level tokens
  foreach occurrence $\mathcal{A}_{ij}$ in Δ do **initialize**($\mathit{this}.C_{ij}$)

$T@Y$ **output**($\mathrm{Cfg}\langle \mathcal{A}, T\rangle$ *this*) :
  foreach $q \in Q_F$ do // iterate over final states
    if ($\mathit{this}.C_0(q)$ is defined) then return $\mathit{this}.C_0(q)$
  return nil

start($\mathrm{Cfg}\langle \mathcal{A}, T\rangle$ *this*, $T\ s$) : // precondition: $[\![\mathit{this}.C]\!]$ and $\langle\!\langle \mathcal{A} \rangle\!\rangle_T(s)$ are disjoint
  $\mathrm{Map}\langle Q, T@Y\rangle\ C_0^{\mathrm{new}} := \bot$
  foreach $q \in Q_I$ do $C_0^{\mathrm{new}}(q) := s.\mathsf{push}(I(q))$ // place token on each initial state
  foreach transition $(p, \mathrm{op}(\mathcal{A}_{i1}, \ldots, \mathcal{A}_{in}), q)$ in Δ do // restart subautomata
    if ($C_0^{\mathrm{new}}(p) \neq \mathsf{nil}$) then // check if there is token on invocation state
      start($\mathit{this}.C_{i1}, C_0^{\mathrm{new}}(p)$); **start**($\mathit{this}.C_{ij}, []$) for all $j = 2, \ldots, n$
      $C_0^{\mathrm{new}}(p) := \mathsf{nil}$
  $\mathit{this}.C_0 := \mathit{this}.C_0 \sqcup C_0^{\mathrm{new}}$

next($\mathrm{Cfg}\langle \mathcal{A}, T\rangle$ *this*, $D\ d$) :
  $\mathrm{Map}\langle Q, T@Y\rangle\ C_0^{\mathrm{next}} := \bot$
  foreach transition (p, ϕ, σ, q) in Δ do
    if $\phi(d) = \mathit{true}$ then
      $T@Y\ s := \mathit{this}.C(p)$ // current stack
      $Y\ y := s.\mathsf{top} \otimes \sigma(d)$ // new value
      $C_0^{\mathrm{next}}(q) := s.\mathsf{pop}.\mathsf{push}(y)$ // new stack
  foreach transition $(p, \mathrm{op}(\mathcal{A}_{i1}, \ldots, \mathcal{A}_{in}), q)$ in Δ do // propagate d to subautomata, collect outputs
    next($\mathit{this}.C_{ij}, d$) for all $j = 1, 2, \ldots, n$
    $T@[Y, Z_1]\ s_1 :=$ **output**($\mathit{this}.C_{i1}$)
    if ($s_1 \neq \mathsf{nil}$) then
      $z_1 := s_1.\mathsf{top}$; $s_1' := s_1.\mathsf{pop}$; $y := s_1'.\mathsf{top}$
      $z_j :=$ **output**($\mathit{this}.C_{ij}$).top for all $j = 2, \ldots, n$
      $C_0^{\mathrm{next}}(q) := s_1'.\mathsf{pop}.\mathsf{push}(y \otimes \mathrm{op}(z_1, z_2, \ldots, z_n))$
  foreach transition $(p, \mathrm{op}(\mathcal{A}_{i1}, \ldots, \mathcal{A}_{in}), q)$ in Δ do // restart subautomata
    if ($C_0^{\mathrm{next}}(p) \neq \mathsf{nil}$) then // check if there is token on invocation state
      start($\mathit{this}.C_{i1}, C_0^{\mathrm{next}}(p)$); **start**($\mathit{this}.C_{ij}, []$) for $j = 2, \ldots, n$
      $C_0^{\mathrm{next}}(p) := \mathsf{nil}$
  $\mathit{this}.C_0 := C_0^{\mathrm{next}}$

Figure 7 General evaluation algorithm for an SA of rank strictly greater than 0.

▶ **Theorem 5.** *Any streaming algorithm for* MinAvgSuffix *requires* $\Omega(n)$ *bits of memory, where* n *is the size of the stream seen so far.*

The following theorem states that the parallel-consistency requirement is essential for evaluation that is quadratic in the size of the automaton. The idea is based on [21].

▶ **Theorem 6.** *The evaluation problem for unambiguous streaming automata without the parallel-consistency restriction requires space exponential in the size of the automaton.*

5 Conclusion

We have considered symbolic weighted automata extended with two crucial features for expressing streaming computations: hierarchical nesting of several aggregators, and parallel execution. The following table summarizes the space complexity of the evaluation problem, where m is the size of the automaton and n the length of the data stream:

	no nesting	nesting without parallelism	consistent parallelism	general parallelism
unambiguous nondeterminism	$O(m)$	$O(m^2)$	$O(m^2)$ [Thm 4]	$O(\exp(m))$ [Thm 6]
general nondeterminism	$O(m)$	$\Omega(n)$ [Thm 5]	$\Omega(n)$	$\Omega(n)$

In *nesting without parallelism*, a transition may call a single subautomaton. *General parallelism* allows transitions with the construct $\mathrm{op}(\mathcal{A}_1, \ldots, \mathcal{A}_m)$, which matches only those strings accepted by every $\mathcal{A}_i$. *Consistent parallelism* restricts this to require $L(\mathcal{A}_1) = \cdots = L(\mathcal{A}_m)$. These complexities assume that the types of the signature require unit space, and that the operations and predicates require unit time.

References

1. Alfred V. Aho. Algorithms for finding patterns in strings. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 5, pages 255–300. MIT Press/Elsevier, 1990.
2. Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? In Tevfik Bultan and Pao-Ann Hsiung, editors, *Proceedings of the 9th International Symposium on Automated Technology for Verification and Analysis (ATVA '11)*, pages 482–491. Springer Berlin Heidelberg, 2011.
3. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
4. Rajeev Alur, Loris D'Antoni, Jyotirmoy Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '13)*, pages 13–22, 2013.
5. Rajeev Alur, Dana Fisman, and Mukund Raghothaman. Regular programming for quantitative properties of data streams. In *Proceedings of the 25th European Symposium on Programming (ESOP '16)*, pages 15–40, 2016.
6. Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '04)*, pages 286–296, 2004.
7. Gerard Berry and Ravi Sethi. From regular expressions to deterministic automata. *Theoretical Computer Science*, 48:117–126, 1986.
8. Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*, 12(4):27:1–27:26, 2011.
9. Ronald Book, Shimon Even, Sheila Greibach, and Gene Ott. Ambiguity in graphs and expressions. *IEEE Transactions on Computers*, C-20(2):149–153, 1971.
10. Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic (TOCL)*, 11(4):23, 2010.
11. Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Nested weighted automata. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '15)*, pages 725–737, 2015.
12. Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Quantitative monitor automata. In Xavier Rival, editor, *Proceedings of the 23rd International Symposium on Static Analysis (SAS '16)*, pages 23–38. Springer Berlin Heidelberg, 2016.
13. S. Chintapalli, D. Dagit, B. Evans, R. Farivar, T. Graves, M. Holderbaugh, Z. Liu, K. Nusbaum, K. Patil, B. J. Peng, and P. Poulosky. Benchmarking streaming computation engines: Storm, Flink and Spark Streaming. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1789–1792, 2016.

14 Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):15:1–15:62, 2012.

15 Loris D'Antoni and Margus Veanes. Equivalence of extended symbolic finite transducers. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV '13)*, pages 624–639, 2013.

16 Loris D'Antoni and Margus Veanes. Minimization of symbolic automata. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14)*, pages 541–553, 2014.

17 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002.

18 Stéphane Demri and Ranko Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic (TOCL)*, 10(3):16:1–16:30, 2009.

19 Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. Springer, 2009.

20 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

21 Martin Fürer. The complexity of the inequivalence problem for regular expressions with intersection. In *Proceedings of the 7th International Colloquium on Automata, Languages and Programming (ICALP '80)*, pages 234–245, 1980.

22 Namit Jain, Shailendra Mishra, Anand Srinivasan, Johannes Gehrke, Jennifer Widom, Hari Balakrishnan, Uğur Çetintemel, Mitch Cherniack, Richard Tibbetts, and Stan Zdonik. Towards a streaming SQL standard. *Proceedings of the VLDB Endowment*, 1(2):1379–1390, 2008.

23 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.

24 Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4(3):405–425, 1994.

25 Konstantinos Mamouras, Mukund Raghothaman, Rajeev Alur, Zachary G. Ives, and Sanjeev Khanna. Streamqre: modular specification and efficient evaluation of quantitative queries over streaming data. In Albert Cohen and Martin T. Vechev, editors, *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18-23, 2017*, pages 693–708. ACM, 2017. doi:10.1145/3062341.3062369.

26 Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

27 J. Ian Munro and Michael S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315–323, 1980.

28 Shanmugavelayutham Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005.

29 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic (TOCL)*, 5(3):403–435, 2004.

30 Richard Edwin Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985.

31 Ken Thompson. Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968.

32 Pete Tucker, Kristin Tufte, Vassilis Papadimos, and David Maier. NEXMark: A benchmark for queries over data streams. Available at `http://datalab.cs.pdx.edu/niagara/NEXMark/`, 2002.

33 Margus Veanes, Peli de Halleux, and Nikolai Tillmann. Rex: Symbolic regular expression explorer. In *Proceedings of the 3rd International Conference on Software Testing, Verification and Validation (ICST '10)*, pages 498–507. IEEE, 2010.

34 Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjorner. Symbolic finite state transducers: Algorithms and applications. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '12)*, pages 137–150, 2012.

On Reversible Transducers*†

Luc Dartois[1], Paulin Fournier[2], Ismaël Jecker[3], and Nathan Lhote[4]

1 **Université Libre de Bruxelles, Brussels, Belgium**
ldartois@ulb.ac.be
2 **Université de Bordeaux, LaBRI, Bordeaux, France**
paulin.fournier@labri.fr
3 **Université Libre de Bruxelles, Brussels, Belgium**
ijecker@ulb.ac.be
4 **Université Libre de Bruxelles, Brussels, Belgium; and Université de Bordeaux, LaBRI, Bordeaux, France**
nlhote@labri.fr

Abstract

Deterministic two-way transducers define the robust class of regular functions which is, among other good properties, closed under composition. However, the best known algorithms for composing two-way transducers cause a double exponential blow-up in the size of the inputs. In this paper, we introduce a class of transducers for which the composition has polynomial complexity. It is the class of reversible transducers, for which the computation steps can be reversed deterministically. While in the one-way setting this class is not very expressive, we prove that any two-way transducer can be made reversible through a single exponential blow-up. As a consequence, we prove that the composition of two-way transducers can be done with a single exponential blow-up in the number of states.

A uniformization of a relation is a function with the same domain and which is included in the original relation. Our main result actually states that we can uniformize any non-deterministic two-way transducer by a reversible transducer with a single exponential blow-up, improving the known result by de Souza which has a quadruple exponential complexity. As a side result, our construction also gives a quadratic transformation from copyless streaming string transducers to two-way transducers, improving the exponential previous bound.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Transducers, reversibility, two-way, uniformization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.113

1 Introduction

Automata and transducers. Automata theory is a prominent domain of theoretical computer science, initiated in the 60s [4] and still very active nowadays. Many extensions of finite automata have been studied such as automata over more complex structures (infinite words, trees, *etc*) or transducers which can be seen as automata with an additional write-only output tape and which will be the focus of our study in the remainder of this article.

Transducers have been studied for almost as long as automata [1] and important results have been obtained, however the theory of transducers is not as advanced as automata theory.

* The full version of this article can be found at [5], http://arxiv.org/abs/1702.07157.
† This work was partially supported by the French ANR project *ExStream* (ANR-13-JS02-0010), the ARC project *Transform* (Fédération Wallonie Bruxelles) and the FNRS PDR project *Flare*. I. Jecker is an F.R.S.-FNRS Aspirant fellow.

Figure 1 The language A^*aA^* can be recognized by a deterministic (left) or codeterministic (right) automaton, but not by a reversible one.

One of the reasons for this is that many descriptions which are equivalent for automata become different in expressiveness in the case of transducers. For instance, deterministic and non-deterministic automata recognize the same class of languages, the *regular languages*. However this is not the case for transducers since in particular a deterministic transducer must realize a function while a non-deterministic one may realize a relation. Similarly, by allowing the reading head to move left and right, one gets a two-way model of automata and it is known that two-way automata are as expressive as one-way automata [12]. However two-way transducers can model relations and functions that are unobtainable in the one-way case, such as the function mirror which reverses its input. Recently, two-way transducers were also proven to be equivalent to the one-way deterministic model of *streaming string transducers* [2], which can be thought of as transducers with write-only registers.

Reversible transducers. A transition system is called *reversible* when for every input, the directed graph of configurations is composed of nodes of in-degree and out-degree at most one. This property is stronger than the more studied notion of determinism since it allows to navigate back and forth between the steps of a computation. In this article, we study the class of transducers that are simultaneously deterministic and codeterministic, *i.e.* reversible. The main motivation for the definition of this class is its good properties with respect to composition. When we consider one-way transducers, runs only go forward and thus determinism gives good properties for composition: the next step of a run is computed in constant time. However, when considering composition of two-way transducers, the second machine can move to the left, which corresponds to rewinding the run of the first machine. Then the stronger property of reversibility allows for this back and forth navigation over runs of transducers, and we recover the property of reaching the next (or previous here) step of a computation in a constant time. This leads to the recovery of the polynomial state complexity of composition which exists for deterministic one-way transducers.

Let us now discuss the expressiveness of reversible transducers. Regarding automata in the one-way case, it is well-known that any regular language can be recognized by a deterministic one-way automaton and symmetrically by a codeterministic one-way automaton, since the mirror of a regular language is still regular. However, the class of one-way reversible automata is very restrictive (see Figure 1 for an example or [11] for a study of its expressive power, where they are called bideterministic). It turns out however, that if we allow bidirectionality then any regular language can be recognized by a reversible automaton. In fact, a two-way reversible automaton can be constructed from either a one-way or two-way automaton using only a linear number of states (see [9] and [10], respectively). We prove, as a consequence of our main theorem, that reversible transducers are as expressive as functional two-way transducers, and exactly capture the class of regular functions. As stated earlier, regular functions are also characterized by streaming string transducers (SST). As a byproduct, we also give a quadratic construction from copyless SST to reversible transducers, improving results from [3, 6].

Synthesis problem and uniformization of transducers. In the bigger picture of verification, two-way transducers can be used to model transformations of programs or non-reactive systems. If we consider the synthesis problem, where the specification is given as a relation of admissible input-output pairs, an implementation is then given as a function, with the same domain, relating a unique output to a given input. The uniformization problem asks if given a relation, we can extract a function that has the same domain, and is included in the relation. We argue that the synthesis problem can be instantiated in the setting of transformations as the problem of uniformization of a non-deterministic two-way transducer by a functional transducer. Our main result states that we can uniformize any non-deterministic two-way transducer by a reversible transducer with a single exponential blow-up.

Related work. As stated earlier, reversible one-way automata were already considered in [11]. Two-way reversible automata were shown to capture the regular languages in [9] by a construction from a one-way deterministic automaton to a two-way reversible automaton with a linear blow-up. This construction was extended to two-way automata in [10], still with a linear complexity. However, these constructions for automata cannot be simply extended to transducers because more information is needed in order to produce the outputs at the right moment. To the best of our knowledge, reversible transducers have not been studied yet, however, since we introduce reversible transducers as a tool for the composition of transducers, our work can be linked with the construction of Hopcroft and Ullman that gives the composition of a one-way transducer and a two-way transducer, while preserving determinism. Our construction strictly improves theirs, since ours produces, with a polynomial complexity instead of an exponential one, a reversible transducer that can in turn be easily composed.

A procedure for the uniformization of a two-way non-deterministic transducer by a deterministic one has been known since [7]. The complexity of this procedure is quadruply exponential, while our construction is done in a single one, and produces a reversible transducer.

Organization of the paper. Preliminary definitions are given in the next Section. In Section 3, we present our main results on composability and expressiveness of reversible transducers. Section 4 is devoted to the main technical construction of the paper. Connections with streaming string transducers are discussed in Section 5 while further works are considered in Section 6.

2 Automata and transducers

Given a finite alphabet A, we denote by A^* the set of finite words over A, and by ε the empty word. We will denote by $A_{\vdash\dashv}$ the alphabet $A \uplus \{\vdash, \dashv\}$, where the new symbols $\vdash$ and $\dashv$ are called the *left* and *right endmarkers*. A *language* over A is a subset $\mathcal{L}$ of A^*. Given two finite alphabets A and B, a *transduction* from A to B is a relation $\mathcal{R} \subseteq A^* \times B^*$.

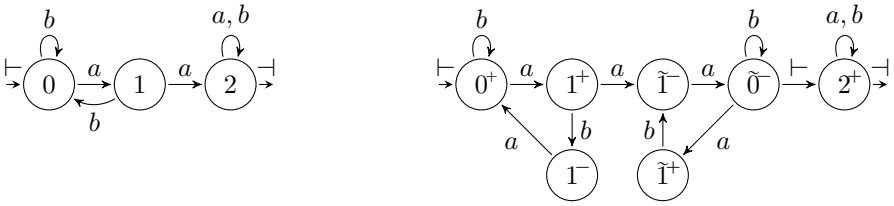
Automata. A *two-way finite state automaton* (2FA) is a tuple $\mathcal{A} = (A, Q, q_I, q_F, \Delta)$, where A is a finite alphabet; Q is a finite set of states partitioned into the set of forward states Q^+ and the set of backward states Q^-; $q_I \in Q^+$ is the initial state; $q_F \in Q^+$ is the final state; $\Delta \subseteq Q \times A_{\vdash\dashv} \times Q$ is the state transition relation. By convention, q_I and q_F are the only forward states verifying $(q_I, \vdash, q) \in \Delta$ and $(q, \dashv, q_F) \in \Delta$ for some $q \in Q$. However, for any backward state $p^- \in Q^-$, Δ might contain transitions $(p^-, \vdash, q)$ and $(q, \dashv, p^-)$, for some

$q \in Q$. Note that, in our figures, we do not represent explicitly the initial and final states, and rather use arrows labeled with the endmarkers to indicate the corresponding transitions. A configuration $u.p.u'$ of $\mathcal{A}$ is composed of two words $u, u' \in A^*_{\vdash\dashv}$ and a state $p \in Q$. The configuration $u.p.u'$ admits a set of successor configurations, defined as follows. If $p \in Q^+$, the input head currently reads the first letter of the suffix $u' = a'v'$. The successor of $u.p.u'$ after a transition $(p, a', q) \in \Delta$ is either $ua'.q.v'$ if $q \in Q^+$, or $u.q.u'$ if $q \in Q^-$. Conversely, if $p \in Q^-$, the input head currently reads the last letter of the prefix $u = va$. The successor of $u.p.u'$ after $(p, a', q) \in \Delta$ is $u.q.u'$ if $q \in Q^+$, or $v.q.au'$ if $q \in Q^-$. For every word $u \in A^*_{\vdash\dashv}$, a *run* of $\mathcal{A}$ on u is a sequence of successive configurations $\varrho = u_0.q_0.u'_0, \dots, u_m.q_m.u'_m$ such that for every $0 \leqslant i \leqslant m$, $u_i u'_i = u$. The run ϱ is called *initial* if it starts in configuration $q_I.u$, *final* if it ends in configuration $u.q_F$, *accepting* if it is both initial and final, and *end-to-end* if it starts and ends on the boundaries of u. More precisely, it is called *left-to-right* if $q_0, q_m \in Q^+$ and $u_0 = u'_m = \varepsilon$; *right-to-left* if $q_0, q_m \in Q^-$ and $u'_0 = u_m = \varepsilon$; *left-to-left* if $q_0 \in Q^+$, $q_m \in Q^-$ and $u_0 = u_m = \varepsilon$; *right-to-right* if $q_0 \in Q^-$, $q_m \in Q^+$ and $u'_0 = u'_m = \varepsilon$. Abusing notations, we also denote by Δ the extension of the state transition relation to a subset of $Q \times A^*_{\vdash\dashv} \times Q$ composed of the triples (p, u, q) such that there exists an end-to-end run on u between p and q. For every triple $(p, u, q) \in \Delta$, we say that q is a *u-successor* of p and that p is a *u-predecessor* of q. The language $\mathcal{L}_\mathcal{A}$ *recognized* by $\mathcal{A}$ is the set of words $u \in A^*$ such that $\vdash u \dashv$ admits an accepting run, i.e., $(q_I, \vdash u \dashv, q_F) \in \Delta$. The automaton $\mathcal{A}$ is called

- a *one-way finite state automaton* (1FA) if the set Q^- is empty;
- *deterministic* if for all $(p, a) \in Q \times A_{\vdash\dashv}$, there is at most one $q \in Q$ verifying $(p, a, q) \in \Delta$;
- *codeterministic* if for all $(q, a) \in Q \times A_{\vdash\dashv}$, there is at most one $p \in Q$ verifying $(p, a, q) \in \Delta$;
- *reversible* if it is both deterministic and codeterministic.
- *weakly branching* if for all $a \in A$ there is at most one state $p \in Q$ and one pair of distinct states $q_1, q_2 \in Q$ such that $(p, a, q_1) \in \Delta$ and $(p, a, q_2) \in \Delta$.

An automaton with several initial and final states can be simulated by using non-determinism while reading the endmarker $\vdash$ and non-codeterminism while reading the endmarker $\dashv$, hence requiring a single initial state and a single final state does not restrict the expressiveness of our model. Let us remark that unlike in the case of most two-way machines, a reversible two-way automaton always halts on any input. Indeed, codeterminism insures that it is never the case that two transitions head to the same configuration. Hence, the unique run (due to determinism) cannot loop since no configuration can be visited twice, and the first configuration starts from the left of the initial endmarker, which is a configuration that cannot be reached later on in the run.

Transducers. A *two-way finite state transducer* (2FT) is a tuple $\mathcal{T} = (A, B, Q, q_I, q_F, \Delta, \mu)$, where B is a finite alphabet; $\mathcal{A}_\mathcal{T} = (A, Q, q_I, q_F, \Delta)$ is a 2FA, called the *underlying automaton* of $\mathcal{T}$; and $\mu : \Delta \to B^*$ is the output function. A run of $\mathcal{T}$ is a run of its underlying automaton, and the language $\mathcal{L}_\mathcal{T}$ recognized by $\mathcal{T}$ is the language $\mathcal{L}_{\mathcal{A}_\mathcal{T}} \in A^*$ recognized by its underlying automaton. Given a run ϱ of $\mathcal{T}$, we set $\mu(\varrho) \in B^*$ as the concatenation of the images by μ of the transitions of $\mathcal{T}$ occurring along ϱ. Note that in the deterministic (or codeterministic) case we are able to extend μ to end-to-end runs since in this case we can unambiguously associate an end-to-end run to a unique sequence of transitions (p, u, q). The transduction $\mathcal{R}_\mathcal{T} \subseteq A^* \times B^*$ defined by $\mathcal{T}$ is the set of pairs (u, v) such that $u \in \mathcal{L}_\mathcal{T}$ and $\mu(\varrho) = v$ for an accepting run ϱ of $\mathcal{A}_\mathcal{T}$ on $\vdash u \dashv$. Two transducers are called *equivalent* if they define the same transduction. A transducer $\mathcal{T}$ is respectively called one-way (1FT), deterministic, weakly branching, codeterministic or reversible, if its underlying automaton has the corresponding property.



(a) A deterministic 1FA $\mathcal{A}_1$ **(b)** A reversible 2FA $\mathcal{A}_2$

Figure 2 Two automata recognizing the language $\mathcal{L}_{aa} = A^*aaA^*$.

Examples. Let us consider the language $\mathcal{L}_{aa} \subseteq \{a, b\}^*$ composed of the words that contain two a symbols in a row. This language is recognized by the deterministic one-way automaton $\mathcal{A}_1$, represented in Figure 2a, and by the reversible two-way automaton $\mathcal{A}_2$, represented in Figure 2b. However, it is not recognizable by a one-way reversible automaton. Let us analyze the behavior of $\mathcal{A}_2$ to see how moving back and forth through the input allows it to recognize $\mathcal{L}_{aa}$ in a reversible manner. First, $\mathcal{A}_2$ uses an intermediate step to go from 1^+ back to 0^+ when reading a b, to avoid creating non-codeterminism. Second, once $\mathcal{A}_2$ reads two consecutive a symbols, it does not go directly in the final state looping on every input, since this would generate non-codeterminism. Instead, $\mathcal{A}_2$ goes in an inverse copy of the first three states, where it rewind its run until the left endmarker. It is then free to go in the looping accepting state.

3 Results on reversible transducers

In this section, we present the main results of our paper. In Subsection 3.1, we show the polynomial composition of reversible transducers. In the following, we give expressiveness results of the class of reversible transducers, relying on this composition procedure as well as the construction presented in Section 4.

3.1 Composition of reversible transducers

The nicest feature of reversible transducers has to be the low complexity (and simplicity) of their composition. Indeed the composition of two such transducers is polynomial in the number of states of the inputs, and the construction is itself quite simple. This is due to the fact that the difficult part in the composition of transducers is to be able to navigate the run easily. In the one-way case, the composition is easy since runs can only move forward. In the two-way case, one needs to advance in the run, but also rewind it. Since the former is made easy by the determinism, and the latter is symmetrically handled by the codeterministim, composition of reversible transducers is straightforward. Let us also remark that only the first transducer has to be reversible in order to obtain a polynomial complexity. However the reversible nature of the obtained transducer depends on the input transducers being both reversible.

▶ **Theorem 1.** *Let $\mathcal{T}_1$ be a reversible two-way transducers and $\mathcal{T}_2$ be two-way transducer with n_1 and n_2 states respectively, such that $\mathcal{T}_1$ can be composed with $\mathcal{T}_2$. Then one can construct a two-way transducer $\mathcal{T}_3$ with $n_1 \cdot n_2$ states realizing $\mathcal{R}_{\mathcal{T}_2} \circ \mathcal{R}_{\mathcal{T}_1}$.*
Furthermore, if $\mathcal{T}_2$ is reversible, deterministic or codeterministic, then so is $\mathcal{T}_3$.

Proof. Let $\mathcal{T}_1 = (A, B, Q, q_I, q_F, \Delta, \mu)$ and $\mathcal{T}_2 = (B, C, P, p_I, p_F, \Gamma, \nu)$. We define $\mathcal{T}_3 = (A, C, Q \times P, (q_I, p_I), (q_F, p_F), \Theta, \xi)$. The idea is that at each step, $\mathcal{T}_3$ simulates a transition

$\delta \in \Delta$, plus the behavior of $\mathcal{T}_2$ over the production $\mu(\delta) \in B^*$ of this transition. To be precise, the transducer $\mathcal{T}_3$ also detects when it simulates an initial or final configuration of $\mathcal{T}_1$ (i.e. upon reading an endmarker in the initial or final state), and accordingly adds the corresponding endmarker to the production before simulating $\mathcal{T}_2$. The partition of the set of states of $\mathcal{T}_3$ depends on the combination of the signs of both components. If $\mathcal{T}_2$ is moving to the right, we use the determinism of $\mathcal{T}_1$, we update the first component of the current state according to the unique transition δ originating from it, and we simulate $\mathcal{T}_2$ entering $\mu(\delta)$ from the left. To do so, $\mathcal{T}_3$ needs to have access to the same letter of the input tape as $\mathcal{T}_1$. Thus, we have $(Q^+ \times P^+) \subseteq (Q \times P)^+$ and $(Q^- \times P^+) \subseteq (Q \times P)^-$. If $\mathcal{T}_2$ is moving to the left, then we use the codeterminism of $\mathcal{T}_1$ to rewind the corresponding run, we update the first component of the current state according to the unique transition δ arriving in it, and we simulate $\mathcal{T}_2$ entering $\mu(\delta)$ from the right. To do so, $\mathcal{T}_3$ needs to have access to the letter on the other side of the input head (with respect to $\mathcal{T}_1$). Thus, we have $(Q^- \times P^-) \subseteq (Q \times P)^+$ and $(Q^+ \times P^-) \subseteq (Q \times P)^-$.

We now define the transition function Θ and the production function ξ. Let $(q, a, q') \in \Delta$ be a transition of $\mathcal{T}_1$ such that $\varrho = (p, v, p')$ is an end-to-end run of $\mathcal{T}_2$, where v denotes the word $\mu(q, a, q') \in B^*$.

- If ϱ is a left-to-right run of $\mathcal{T}_2$, then $((q, p), a, (q', p'))$ belongs to Θ and produces $\nu(p, v, p')$.
- If ϱ is a left-to-left run of $\mathcal{T}_2$, then $((q, p), a, (q, p'))$ belongs to Θ and produces $\nu(p, v, p')$.
- If ϱ is a right-to-right run of $\mathcal{T}_2$, then $((q', p), a, (q', p'))$ belongs to Θ and produces $\nu(p, v, p')$.
- If ϱ is a right-to-left run of $\mathcal{T}_2$, then $((q', p), a, (q, p'))$ belongs to Θ and produces $\nu(p, v, p')$.

The behavior of the transducer $\mathcal{T}_3$ is completely determined by the combined behaviors of transducers $\mathcal{T}_1$ and $\mathcal{T}_2$. When $\mathcal{T}_3$ simulates a transition of $\mathcal{T}_1$, it also simulates the corresponding end-to-end run of $\mathcal{T}_2$ over the production of this transition. If the direction of both simulations is the same, then $\mathcal{T}_3$ moves forward. Otherwise, it moves backward. The transducer stops when it reaches a final state in both $\mathcal{T}_1$ over the input, and $\mathcal{T}_2$ over the simulated run over partial productions of the run of $\mathcal{T}_1$ over the input. Then the final output of $\mathcal{T}_3$ is the concatenation of the outputs of the partial runs of $\mathcal{T}_2$ it simulates, which corresponds to the output of $\mathcal{T}_1$. Hence, the transducer $\mathcal{T}_3$ realizes the composition $\mathcal{T}_2 \circ \mathcal{T}_1$. The possible determinism (resp. codeterminism) of $\mathcal{T}_3$ is a direct consequence of the one of $\mathcal{T}_1$ and $\mathcal{T}_2$. Indeed, a witness of non-determinism (resp. non codeterminism) of $\mathcal{T}_3$ can be traced back to a witness run of either $\mathcal{T}_1$ or $\mathcal{T}_2$ that is not deterministic (resp.codeterministic). ◀

3.2 One-way transducers

In the next subsections, we give some procedures to construct a reversible transducer from either a one-way or a two-way transducer. The main ingredient of the proofs is the technical construction from Lemma 6 (presented in Section 4) which constructs a reversible transducer from a *weakly branching* codeterministic one-way transducer. The proofs of this section share the same structure: in order to build a reversible transducer that defines a function $\mathcal{F}$, we express $\mathcal{F}$ as a composition of transductions definable by reversible transducers, and we conclude by using Theorem 1. The detailed constructions are presented in the full version of this article. Building on Lemma 6, we show that codeterministic one-way transducers can be expressed as the composition of weakly branching codeterministic ones.

▶ **Theorem 2.** *Given a codeterministic 1FT with n states, one can effectively construct an equivalent reversible 2FT with $4n^2$ states.*

Proof. Let $\mathcal{T}$ be a codeterministic 1FT with n states. The function $\mathcal{R}_\mathcal{T}$ can be expressed as the composition $\mathcal{R}_{\mathcal{T}'} \circ \mathcal{R}_\mathcal{M}$, where $\mathcal{M}$ and $\mathcal{T}'$ are defined as follows.

- Transducer $\mathcal{M}$ is a reversible 1FT with a single state that multiplies all the letters of the input word by n while marking them with a state of $\mathcal{T}$;
- Transducer $\mathcal{T}'$ is a weakly branching and codeterministic one-way transducer that has the same set of states as $\mathcal{T}$. On input $\mathcal{R}_\mathcal{M}(u)$, $\mathcal{T}'$ mimics the behavior of $\mathcal{T}$ on u, while using the fact that the input word is larger to desynchronize the non-deterministic branchings that were occurring simultaneously in $\mathcal{T}$. Intuitively, a transition of $\mathcal{T}$ can only be taken by $\mathcal{T}'$ at the copy of the letter corresponding to the target state of the transition.

By Lemma 6, $\mathcal{T}'$ can be made into a reversible 2FT $\mathcal{T}''$ with $4n^2$ states. Therefore, since both $\mathcal{T}''$ and $\mathcal{M}$ are reversible, we can conclude using Theorem 1, finally obtaining a reversible 2FT with $4n^2$ states equivalent to $\mathcal{T}$. ◀

Using composition again, the statement can be extended to deterministic one-way transducers.

▶ **Theorem 3.** *Given a deterministic 1FT with n states, one can effectively construct an equivalent reversible 2FT with $36n^2$ states.*

Proof. Let $\mathcal{T}$ be a deterministic 1FT with n states. Then $\overline{\mathcal{T}}$, the transducer obtained by reversing all transitions of $\mathcal{T}$, is codeterministic. The function $\mathcal{R}_\mathcal{T}$ can be expressed as the composition $\mathcal{R}_{M_B} \circ \mathcal{R}_{\overline{\mathcal{T}}} \circ \mathcal{R}_{M_A}$, where M_A and M_B realize the mirror functions over the input and output alphabet of $\mathcal{T}$ respectively. Both of them are realized by a 3 states reversible transducer. Then by Theorem 2, we can construct $\overline{\mathcal{T}}'$ which has $4n^2$ states, is reversible and realizes the same function as $\overline{\mathcal{T}}$. By Theorem 1, we can compose the three transducers, finally obtaining a reversible transducer equivalent to $\mathcal{T}$ with $9 \cdot 4n^2$ states. ◀

3.3 Two-way transducers

We now prove our main result, which states that any two-way transducer can be uniformized by a reversible two-way transducer. Let us recall that uniformization by a deterministic transducer was done in [7]. We use similar ideas for the uniformization. The key difference is that we rely on the construction of Section 4 while in [7], the main construction is the tree-trimming construction of Hopcroft-Ullman from [8].

▶ **Theorem 4.** *Given a 2FT $\mathcal{T}$ with n states, one can effectively construct a reversible 2FT $\mathcal{T}'$ whose number of states is exponential in n, and verifying $\mathcal{L}_{\mathcal{A}_{\mathcal{T}'}} = \mathcal{L}_{\mathcal{A}_\mathcal{T}}$ and $\mathcal{R}_{\mathcal{T}'} \subseteq \mathcal{R}_\mathcal{T}$.*

Proof. Let $\mathcal{T} = (A, B, Q, q_I, q_F, \Delta, \mu)$ be a 2FT with n states. We define a function uniformizing $\mathcal{R}_\mathcal{T}$ as the composition $\mathcal{R}_{\mathcal{T}'} \circ \mathcal{R}_\mathcal{U} \circ \mathcal{R}_{\mathcal{D}_r}$, where $\mathcal{D}_r$, $\mathcal{U}$ and $\mathcal{T}$ are defined as follows.

- The right-oracle $\mathcal{D}_r$ is a codeterministic one-way transducer with 2^{n^2+n} states that enriches each letter of the input word $u \in A^*_{\vdash\dashv}$ with information concerning the behavior of $\mathcal{T}$ on the corresponding suffix, represented by the set of pairs that admit a left-to-left run, and the set of states from which $\mathcal{T}$ can reach the final state.
- The uniformizer $\mathcal{U}$ is a deterministic one-way transducer with $n!$ states. On input $u' = \mathcal{R}_{\mathcal{D}_r}(u)$, $\mathcal{U}$ uses the information provided by $\mathcal{D}_r$ to pick a run ϱ_u of $\mathcal{T}$ on input u, and enriches each letter a_i of the input word with the sequence of transitions occurring in the run ϱ_u that correspond to the letter a_i.
- Finally, the reversible transducer $\mathcal{T}'$ has the same set of states as $\mathcal{T}$, and follows the instructions left by $\mathcal{U}$ to solve the non-determinism and the non-codeterminism.

As a consequence of Theorem 2 and Theorem 3, there exist two reversible 2FT $\mathcal{D}_r{}'$ and $\mathcal{U}'$ whose number of states are exponential in n, and that verify $\mathcal{R}_{\mathcal{D}_r'} = \mathcal{R}_{\mathcal{D}_r}$ and $\mathcal{R}_{\mathcal{U}'} = \mathcal{R}_{\mathcal{U}}$. Therefore, since $\mathcal{D}_r{}', \mathcal{U}'$ and $\mathcal{T}'$ are reversible, by Theorem 1 there exists a reversible transducer $\mathcal{T}''$ whose number of states is exponential in n, and that satisfies $\mathcal{R}_{\mathcal{T}''} = \mathcal{R}_{\mathcal{T}'} \circ \mathcal{R}_{\mathcal{U}'} \circ \mathcal{R}_{\mathcal{D}_r{}'} = \mathcal{R}_{\mathcal{T}}$. ◀

The following result is a direct corollary of Theorem 4, applied to deterministic two-way transducers.

▶ **Corollary 5.** *Reversible two-way transducers are as expressive as deterministic two-way transducers.*

4 The tree-outline construction

In this section lies the heart of our result. We show that any *weakly branching* and codeterministic transducer can be made reversible. These hypotheses allow us to simplify our proof, and still obtain a more general result, as a corollary.

▶ **Lemma 6.** *Let $\mathcal{T}$ be a codeterministic and weakly branching 1FT with m states. Then one can effectively construct a reversible 2FT $\mathcal{T}'$ with $4m^2$ states that is equivalent to $\mathcal{T}$.*

Proof. The construction presented in this proof is illustrated on an example in Figure 3. Let $\mathcal{T} = (A, Q, q_I, q_F, \Delta, \mu)$ be a codeterministic 1FT, and let $\prec$ be a total order over Q. As an example, take the codeterministic 1FT T presented in figure 3a.

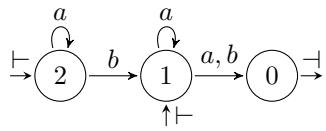
Let $\mathcal{T}' = (A, \mathcal{F}, f_I, f_F, \Delta', \mu')$ be a 2FT defined as follows:

On input $u \in \mathcal{L}_{\mathcal{T}}$, $\mathcal{T}'$ explores depth first the run-tree T_u composed of the initial runs of $\mathcal{T}$ on the word $\vdash u \dashv$ (illustrated in Figure 3b). More precisely it explores the "sheath" of the run-tree (see Figure 3c for a graphical representation). To do this, the states of $\mathcal{T}'$ are composed of two states of $\mathcal{T}$ with a marker. The first state represents the upper part of the sheath, while the second state represents the lower part. Moreover the marker is used to denote whether we are above the branch ($\underline{q}$) or below the branch ($\overline{q}$).

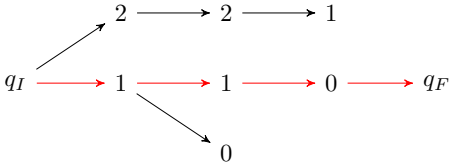
Initially we start with the state $(\underline{q_I}, \overline{q_I})$ and go forward according to the transitions of $\mathcal{T}$. While moving forward whenever a branching state q is reached, if the state is marked $\underline{q}$ it moves to the maximal successor of q (in order to stay above the branch) and symmetrically if the state is marked $\overline{q}$ it moves to the minimal successor of q (in order to stay below the branch). Whenever one of the branches reaches a dead end we continue the sheath exploration by switching the marker (*i.e.* changing from above the branch to below or vice-versa) and start moving backward accordingly to the transitions of $\mathcal{T}$. While moving backward, if the successor of a branching state q is reached, while we were inside the fork, *e.g.* in state $\overline{q_{\max}}$ (where $q_{\max}$ is the maximal successor of q), we continue the exploration of the sheath by going in the state $\underline{q_{\min}}$ and we start moving forward again. Whenever the upper and lower explorations of the sheath coincide, *i.e.* in states of the form $(\underline{q}, \overline{q})$ (represented in red in Figure 3d), it means we are on a prefix of the accepting run, we can thus produce the corresponding output.

Formally $\mathcal{T}' = (A, \mathcal{F}, f_I, f_F, \Delta', \mu')$ is defined as follows:

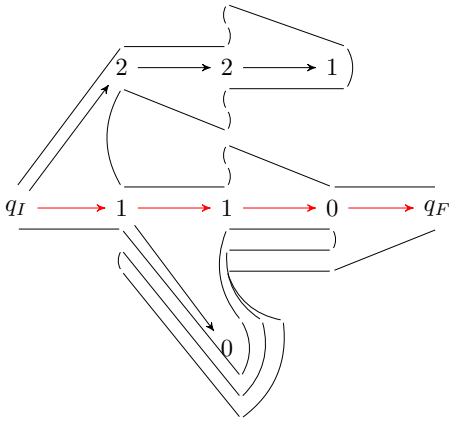
- $\mathcal{F} = \mathcal{F}^+ \cup \mathcal{F}^-$ where $\mathcal{F}^+ = \underline{Q} \times \overline{Q} \cup \overline{Q} \times \underline{Q}$ and $\mathcal{F}^- = (\overline{Q} \times \overline{Q} \cup \underline{Q} \times \underline{Q}) \backslash \{(\overline{p}, \overline{p}), (\underline{p}, \underline{p}) \mid p \in Q\}$
- $f_I = (\underline{q_I}, \overline{q_I})$
- $f_F = (\underline{q_F}, \overline{q_F})$
- We define the transition relation Δ' by differentiating several types of behavior, depending on whether we are going **f**orward, or **b**ackward, whether the **u**pper component or the



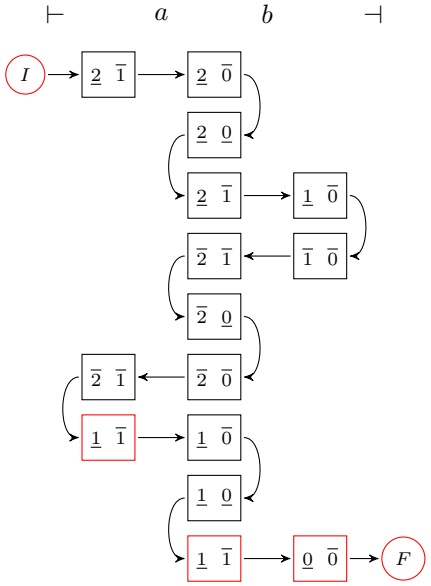
(a) A codeterministic transducer $\mathcal{T}$



(b) The run-tree of $\mathcal{T}$ on $\vdash ab \dashv$



(c) Graphical representation of the run of $\mathcal{T}'$



(d) The run of $\mathcal{T}'$

Figure 3 Illustrations of the proof concepts of Lemma 6. For the sake of clarity, the outputs of $\mathcal{T}$ are omitted.

lower component is involved, and whether it is above or below its branch. Let p and q be two states in Q, and $a \in A$ be a letter.

If p has no a-successor, then:

(fua) $\big((\underline{p}, \overline{q}), a, (\overline{p}, \overline{q})\big) \in \Delta'$, and

(fuw) $\big((\overline{p}, \underline{q}), a, (\underline{p}, \underline{q})\big) \in \Delta'$.

If p has an a-successor, but not q, then:

(flw) $\big((\underline{p}, \overline{q}), a, (\underline{p}, \underline{q})\big) \in \Delta'$, and

(fla) $\big((\overline{p}, \underline{q}), a, (\overline{p}, \overline{q})\big) \in \Delta'$.

Otherwise, p and q admit an a-successor. We denote $p_{\max}$ (resp. $p_{\min}$) the maximal (resp. minimal) a-successor of p (resp. q) with respect to $\prec$. Then:

If $p_{\min} \neq p_{\max}$, then:

(buw) $\big((\overline{p_{\max}}, \overline{q}), a, (\underline{p_{\min}}, \overline{q})\big) \in \Delta'$, and

(bua) $\big((\underline{p_{\min}}, \underline{q}), a, (\overline{p_{\max}}, \underline{q})\big) \in \Delta'$.

If $q_{\min} \neq q_{\max}$, then:

(bla) $\big((\underline{p}, \underline{q_{\min}}), a, (\underline{p}, \overline{q_{\max}})\big) \in \Delta'$

(blw) $\big((\overline{p}, \overline{q_{\max}}), a, (\overline{p}, \underline{q_{\min}})\big) \in \Delta'$

(fualw) $\big((\underline{p}, \overline{q}), a, (\underline{p_{\max}}, \overline{q_{\min}})\big) \in \Delta'$,

(fuwla) $\big((\overline{p}, \underline{q}), a, (\overline{p_{\min}}, \underline{q_{\max}})\big) \in \Delta'$,

(bulw) $\big((\overline{p_{\min}}, \overline{q_{\min}}), a, (\overline{p}, \overline{q})\big) \in \Delta'$, and

(bula) $\big((\underline{p_{\max}}, \underline{q_{\max}}), a, (\underline{p}, \underline{q})\big) \in \Delta'$.

- We define μ' as the function such that for every $(p, a, q) \in \Delta$:
  - if $q = p_{\min} = p_{\max}$ then $\mu'\big((\underline{p}, \overline{p}), a, (\underline{q}, \overline{q})\big) = \mu(p, a, q)$
  - if $q = p_{\min} \neq p_{\max}$ then $\mu'\big((\overline{p_{\max}}, \overline{q}), a, (\underline{q}, \overline{q})\big) = \mu(p, a, q)$
  - if $q = p_{\max} \neq p_{\min}$ then $\mu'\big((\underline{q}, \underline{p_{\min}}), a, (\underline{q}, \overline{q})\big) = \mu(p, a, q)$

  and $\mu'(t) = \varepsilon$ for every $t \in \Delta'$ which is not of one of theses forms.

One can see, by a case study that $\mathcal{T}'$ is deterministic. Indeed, the fact that $\mathcal{T}$ is weakly branching implies that the rules **(buw)** and **(bua)** are mutually exclusive with the rules **(bla)** and **(blw)**. Moreover these four rules are mutually exclusive with the rules **(bulw)** and **(bula)** by construction. And since $\mathcal{T}$ is codeterministic, the predecessor is unique. Finally, the rules **(fua)**, **(fuw)**, **(flw)**, **(fla)**, **(fualw)**, and **(fuwla)** are mutually exclusive by construction, since the conditions on the number of a-successors are incompatible.

A similar case study gives that $\mathcal{T}'$ is codeterministic. Hence $\mathcal{T}'$ is reversible.

A detailed proof of the equivalence between $\mathcal{T}$ and $\mathcal{T}'$ can be found in the full version of the article, and we give a quick intuition of the proof. It relies on two main arguments. The first one is that at any point if the transducer $\mathcal{T}'$ follows two different runs, then it will come back to the same position, where the state that leads to the shortest run has been switched. Following this, we then prove that upon any branching, $\mathcal{T}'$ comes back to the same position but since the shortest run has been switched, it is able to solve the non-determinism, take the transition of the accepting run and produce the correct output. ◀

5 Streaming string transducers

Streaming string transducers, which were introduced in [2], are one-way deterministic automata with additional write-only registers. Partial outputs are stored in the registers via register updates, and at the end of a run an output is produced using these registers. Thus an SST realizes a function over words, and it is known that they are as expressive as 2FT [2]. Direct transformations from SST to 2FT were already considered in [3, 6]. However, these constructions were exponential in the number of states (and linear in the number of registers). Using Theorem 3, we are able to get a construction which is quadratic in the number of states (and also linear in the number of registers). Before explaining the construction, let us formally define the SST.

Substitutions. Given a finite alphabet A and a finite set $\mathcal{X}$ of variables. Let $\mathcal{S}_{\mathcal{X},A}$ denote the set of functions $\sigma : \mathcal{X} \to (\mathcal{X} \cup A)^*$. The elements of $\mathcal{S}_{\mathcal{X},A}$ are called *substitutions*. Any substitution σ can be extended to range over both variables and letters of the output alphabet $\hat{\sigma} : (\mathcal{X} \cup A)^* \to (\mathcal{X} \cup A)^*$ by setting $\hat{\sigma}(a) = a$ for every $a \in A^*$ and $\hat{\sigma}(uv) = \hat{\sigma}(u)\hat{\sigma}(v)$ for $u, v \in (\mathcal{X} \cup A)^*$. This allows us to easily compose substitutions from $\mathcal{S}_{\mathcal{X},A}$ by defining $\sigma_2 \circ \sigma_1$ as the usual function composition $\hat{\sigma_2} \circ \sigma_1$. We denote by $\mathsf{Id}_{\mathcal{X}}$ the identity element of $\mathcal{S}_{\mathcal{X},A}$, which maps every variable to itself, and by σ_ε the substitution mapping every variable to ε.

A substitution σ is called *copyless* if for every $X \in \mathcal{X}$, each variable $Y \in \mathcal{X}$ appears at most once in $\sigma(X)$, and for every $Y \in \mathcal{X}$ there exists at most one $X \in \mathcal{X}$ such that Y appears in $\sigma(X)$.

Streaming string transducers. A *streaming string transducer* (SSTfor short) is a tuple $\mathcal{Z} = (A, B, Q, q_I, q_F, \Delta, \mathcal{X}, O, \tau)$, where B is the output alphabet, $\mathcal{A}_\mathcal{Z} = (A, Q, q_I, q_F, \Delta)$ is a one-way deterministic automaton, called the underlying automaton of $\mathcal{Z}$; $\mathcal{X}$ is a finite set of variables; $O \in \mathcal{X}$ is the final variable; $\tau : \Delta \to \mathcal{S}_{\mathcal{X},B}$ is the output function. A run of $\mathcal{Z}$ is a run of its underlying automaton, and the language $\mathcal{L}_\mathcal{Z}$ recognized by $\mathcal{Z}$ is the language $\mathcal{L}_{\mathcal{A}_\mathcal{Z}} \in A^*$ recognized by its underlying automaton. Given a run ϱ of $\mathcal{Z}$ on u, we set $\tau(\varrho) \in \mathcal{S}_{\mathcal{X},B}$ as the composition of the images by τ of the transitions of $\mathcal{Z}$ occuring along ϱ. The transduction $\mathcal{R}_\mathcal{Z} \subseteq A^* \times B^*$ defined by $\mathcal{Z}$ is the function mapping any word u of $\mathcal{L}_{\mathcal{A}_\mathcal{Z}}$ to $(\sigma_\varepsilon \circ \tau(\varrho))(O)$, where ϱ is the single accepting run of $\mathcal{A}_\mathcal{Z}$ on $\vdash u \dashv$. The SST $\mathcal{Z}$ is called copyless, if for every run ϱ of $\mathcal{Z}$ the substitution $\tau(\varrho)$ is copyless.

▶ **Theorem 7.** *Given a copyless SST with n states and m variables, one can effectively construct an equivalent reversible 2FT with $8m \cdot n^2$ states.*

Proof. We write $\mathcal{Z}$ as the composition of a one-way deterministic transducer $\mathcal{D}_1$ and a reversible one $\mathcal{T}$. The first transducer has the same underlying automaton as $\mathcal{Z}$, the difference being that it outputs the substitution of $\mathcal{Z}$ instead of applying it. Then $\mathcal{T}$ is a transducer that navigates the substitutions to produce the output word of $\mathcal{Z}$. This can be done in a reversible fashion thanks to the property of copylessness of $\mathcal{Z}$. Note that the transducer $\mathcal{T}$ was already defined in [6], Section 4. Formally, let $\mathcal{Z} = (A, B, Q, q_I, q_F, \Delta, \mathcal{X}, O, \tau)$ be a copyless SST with n states and m variables, and let $S_{\mathcal{Z}} \subset \mathcal{S}_{A,\mathcal{X}}$ be the range of τ. We express $\mathcal{R}_{\mathcal{Z}}$ as the composition of $\mathcal{R}_{\mathcal{D}_1} : \mathcal{L}_{\mathcal{A}_{\mathcal{Z}}} \to S_{\mathcal{Z}}^*$ and $\mathcal{R}_{\mathcal{T}_2} : S_{\mathcal{Z}}^* \to B^*$, defined as follows.

- $\mathcal{D}_1$ is a deterministic 1FT obtained by stripping $\mathcal{Z}$ of its SST structure, i.e., $\mathcal{D}_1 = (A, S_{\mathcal{Z}}, Q, q_I, q_F, \Delta, \tau)$. It maps each word of $\mathcal{L}_{\mathcal{A}_{\mathcal{Z}}}$ to the corresponding sequence of substitutions.
- $\mathcal{T} = (S_{\mathcal{Z}}, B, P, init, fin, \Gamma, \nu)$ where $P^+ = \mathcal{X}^o \uplus \{init, fin\}$, $P^- = \mathcal{X}^i$. States labeled by i (resp. o) are *in* (resp. *out*) states and appear when we start (resp. finish) producing a variable. We define Γ and ν as follows:
  - $(init, \sigma, init) \in \Gamma$;
  - $(init, \vdash, O^i) \in \Gamma$;
  - $(O^o, \dashv, fin) \in \Gamma$;
  - $(X^i, \sigma, Y^i) \in \Gamma$ and $\nu((X^i, \sigma, Y^i)) = v$ if $\sigma(X) = vY...$ with $v \in B^*$;
  - $(X^i, \sigma, X^o) \in \Gamma$ and $\nu((X^i, \sigma, X^o)) = v$ if $\sigma(X) = v$;
  - $(X^o, \sigma, Y^i) \in \Gamma$ and $\nu((X^o, \sigma, Y^i)) = v$ if there exists a variable Z where $\sigma(Z) = ...XvY..$;
  - $(X^o, \sigma, Y^o) \in \Gamma$ and $\nu((X^o, \sigma, Y^o)) = v$ if $\sigma(Y) = ...Xv$.

  Due to copylessness, for any σ and any variable X, there is at most one variable Y such that X appears in $\sigma(Y)$. Plus, as the variables are ordered by their appearance in $\sigma(Y)$, the transducer $\mathcal{T}$ is reversible. It starts by reaching the end of the word, then starts producing the variable O. By following the substitution tree of O, it then produces exactly the image of the input by $\mathcal{Z}$.

By Theorem 3, there exists a reversible 2FT $\mathcal{D}_1'$ with $4n^2$ states satisfying $\mathcal{R}_{\mathcal{D}_1'} = \mathcal{R}_{\mathcal{D}_1}$. Finally, since both $\mathcal{D}_1'$ and $\mathcal{T}$ are reversible, by Theorem 1 there exists a reversible transducer $\mathcal{T}'$ with $8m \cdot n^2$ states such that $\mathcal{R}_{\mathcal{T}'} = \mathcal{R}_{\mathcal{T}} \circ \mathcal{R}_{\mathcal{D}_1'} = \mathcal{R}_{\mathcal{Z}}$. ◀

6 Conclusion

We argue that reversible transducers can be seen as a canonical representation of regular transductions. We believe that the polynomial complexity of composition of reversible transducers is a good tool for the verification of cascades of transformations of non-reactive systems. While preserving the expressive power of functional transducers, reversible transducers allow for easier manipulations, the best example being their polynomial composition. Thanks to the tree-outline construction that we presented, one can uniformize a non-determinsitic two-way transducer by a reversible one with a single exponential blow-up. This improves the known constructions that were used up to now, however it is still open whether this blow-up can be avoided. In [10] the authors extended the result of [9] and showed that deterministic two-way automata can be made reversible with a linear blow-up. We conjecture that our approach can also be extended to the two-way case and that deterministic two-way transducers can be made reversible using only a polynomial number of states.

We have shown that applying this construction allows for a quadratic transformation from copyless streaming string transducers to reversible two-way transducers. The converse does not hold, since even on languages deterministic two-way automata are known to be exponentially more succint than deterministic one-way automata. Beyond this, we do not reject the possibility that if one were to embed some recognition power into the variables of a SST, it may be possible to have a polynomial transformation from reversible automata to copyless SST.

References

1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. A general theory of translation. *Mathematical Systems Theory*, 3(3):193–221, 1969.

2 Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, pages 1–12, 2010.

3 Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 65–74, 2012.

4 Julius R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.

5 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On reversible transducers. *CoRR*, abs/1702.07157, 2017. URL: `http://arxiv.org/abs/1702.07157`.

6 Luc Dartois, Ismaël Jecker, and Pierre-Alain Reynier. Aperiodic string transducers. In *Developments in Language Theory – 20th International Conference, DLT 2016, Montréal, Canada, July 25-28, 2016, Proceedings*, pages 125–137, 2016.

7 Rodrigo de Souza. Uniformisation of two-way transducers. In *Language and Automata Theory and Applications – 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, pages 547–558, 2013.

8 John E. Hopcroft and Jeffrey D. Ullman. An approach to a unified theory of automata. In *8th Annual Symposium on Switching and Automata Theory, Austin, Texas, USA, October 18-20, 1967*, pages 140–147, 1967.

9 Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *38th Annual Symposium on Foundations of Computer Science, FOCS'97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 66–75, 1997.

10 Michal Kunc and Alexander Okhotin. Reversibility of computations in graph-walking automata. In *Mathematical Foundations of Computer Science 2013 – 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, pages 595–606, 2013.

11 Jean-Eric Pin. On reversible automata. In *LATIN'92, 1st Latin American Symposium on Theoretical Informatics, São Paulo, Brazil, April 6-10, 1992, Proceedings*, pages 401–416, 1992.

12 John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.

Which Classes of Origin Graphs Are Generated by Transducers?*

Mikołaj Bojańczyk[1], Laure Daviaud[2], Bruno Guillon[3], and Vincent Penelle[4]

1 **University of Warsaw, Warsaw, Poland**
`mbojanczyk@mimuw.edu.pl`
2 **University of Warsaw, Warsaw, Poland**
`ldaviaud@mimuw.edu.pl`
3 **University of Warsaw, Warsaw, Poland**
`guillonb@mimuw.edu.pl`
4 **University of Warsaw, Warsaw, Poland**
`penelle@mimuw.edu.pl`

Abstract

We study various models of transducers equipped with origin information. We consider the semantics of these models as particular graphs, called origin graphs, and we characterise the families of such graphs recognised by streaming string transducers.

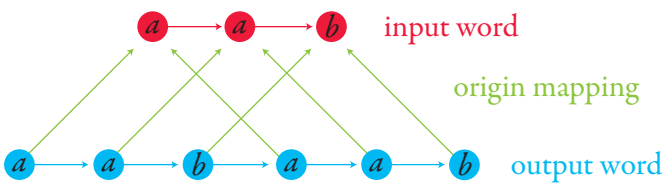
1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Streaming String Transducers, Origin Semantics, String-to-String Transductions, MSO Definability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.114

1 Introduction

This paper is about string-to-string transductions with origin semantics. A string-to-string transduction is a binary relation between strings over fixed input and output alphabets. Examples include the *squaring* transduction $w \mapsto ww$ or the *subword* transduction, which is the set of pairs (u, v) such that v is a subword of u. Note that squaring is a function, while subword is a relation; both types will be studied. The origin semantics of a transduction (technically speaking, of a device computing it) consists not only of pairs (u, v) of input and output words, but also gives an *origin mapping* that specifies which positions of the input word were used to produce which positions of the output word. For example, suppose that we model the squaring transduction by a two-way automaton which does two consecutive left-to-right passes over the input word and copies the input word in each one. In this case, the origin semantics over a particular input word can be visualised as follows:



* This paper is part of LIPA, a project funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 683080).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 114; pp. 114:1–114:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

An object as in the above picture is called an *origin graph*, and we define an *origin string-to-string transduction* to be a set of origin graphs. Origin semantics is more fine-grained semantics than the usual semantics of transductions in the sense that even if two devices compute the same transduction, they might not have the same origin semantics. Origin semantics were introduced in [5], where it was shown that existing models of transducers, such as two-way transducers (also called two-way automata with outputs, e.g. [11]), MSO string-to-string transductions [9], or streaming string transducers [1, 3] can be equipped with origin semantics so that they generate not sets of pairs of words, but sets of origin graphs (the name origin graph is new in this paper). Furthermore, existing results on equivalence between models remain true when the origin semantics are used [5]. We aim to study sets of origin graphs that are origin semantics of transducers. There are two parts.

In the first part, we study decision problems that involve MSO properties of origin graphs. The main result (not very hard) is that when given an MSO formula on origin graphs, and an origin string-to-string transduction realised by a nondeterministic streaming string transducer, one can decide if the formula is true in some origin graph from the transduction. This result gives a generic framework for deciding questions like: is the origin mapping order preserving? The result is proved by using techniques from the theory of MSO transductions [9].

In the second part, we study the structural properties of those classes of origin graphs that can be obtained by taking the origin semantics of some streaming string transducer (or any of the other equivalent models). Our goal is to describe them in a machine independent way. The principal result, Theorem 10, gives the following characterisation: a set of origin graphs is the origin semantics of some nondeterministic streaming string transducer if and only if it has three properties: (1) it is MSO-definable as a set of coloured graphs; (2) it has bounded degree; and (3) it has bounded crossing, which means intuitively that the origin mapping does not oscillate too much. The idea to give a machine independent characterisation of origin semantics was already present in Theorem 1 from [5]. We believe however that origin graphs are a more intuitive and visual notion than the factorised words used in [5]. Furthemore, modelling the origin as a relational structure (the input word, the output word, and the origin information) makes it possible to use MSO logic, or to make the connection with structural notions such as clique width or tree width. Hence, this paper can be seen as a natural complement to the results form [5], or possibly a clearer picture. Furthermore, we study more general models than [5], in particular we allow nondeterminism and ε-transitions.

Important related work is the paper [10], which proposes to use logic to describe properties of origin graphs (they use the name *productions*). In [10], the authors ask about the decidability of checking if a transducer, seen as a set of origin graphs, satisfies a specification given in some logic. This is the direct inspiration for our results in Section 3, in particular our Theorem 6 which says that it is decidable if a given MSO formula is true in some origin graph generated by a given transducer. In [10], the logic used to express properties of origin graphs is a strict fragment of MSO, called $\mathcal{L}_T$, a type of two-variable logic. Therefore, our Theorem 6 is stronger than the model-checking result mentioned in [10, Section VI]. The reason why [10] uses a logic weaker than MSO is that they want to answer different questions than model-checking a given transducer; in particular the logic $\mathcal{L}_T$ is shown to have decidable satisfiability when evaluated on the class of all origin graphs, contrary to MSO.

2 Origin semantics

Define a *string-to-string transduction with input alphabet* Σ *and output alphabet* Γ to be a relation $R \subseteq \Sigma^* \times \Gamma^*$. A transduction is *functional* when it is a partial function.

▶ **Example 1** (Running example). Define the *squaring* functional transduction to be the function $\{a,b\}^* \to \{a,b\}^*$ defined by $w \mapsto ww$.

2.1 Transductions recognised by streaming string transducers

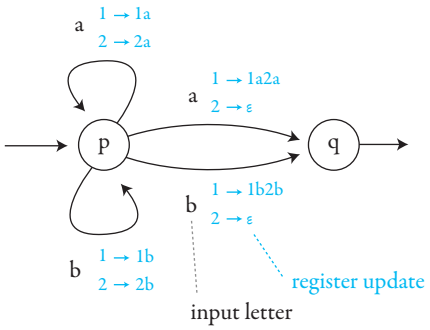
The main topic of this paper is the class of string-to-string transductions recognised by streaming string transducers [1]. Another equivalent presentation of this class is MSO string-to-string transductions, or a suitably defined nondeterministic version of two-way automata with output. We will mainly use the definition in terms of streaming string transducers, so we begin by defining that model.

Streaming string transducers. A streaming string transducer is a device which is used to transform (possibly non-deterministically) a word over an input alphabet into a word over an output alphabet. Because of nondeterminism, one input might produce several outputs, possibly zero. The output is prepared by using registers. Before describing the device itself, let us explain how registers are used. Let Γ be an output alphabet and let $\mathcal{R}$ be a set of *register names*. Define a *register valuation* to be a function $\mathcal{R} \to \Gamma^*$ and a *register update* to be a function $\mathcal{R} \to (\Gamma \cup \mathcal{R})^*$. A register update is viewed as a function from register valuations to register valuations in the following sense: if v is a register valuation and u is a register update, then applying u to v yields a register valuation which stores in register r the value $u(r)$ with each register name replaced by its contents under v. For example if $\mathcal{R}$ has only one register, and u is a register update defined by $r \mapsto ara$, then applying u to a register valuation simply adds the letter a to both the beginning and end of the word stored in the unique register r. A register update u is called *copyless* if for every register name r there is at most one register name s such that r appears in $u(s)$, and furthermore r appears at most once in $u(s)$.

▶ **Definition 2** (Streaming string transducer). The syntax[1] of a streaming string transducer with input alphabet Σ and output alphabet Γ consists of:

- a nondeterministic automaton $\mathcal{B}$ with input alphabet Σ, called the *underlying automaton*;
- a finite set of *register names* $\mathcal{R}$ with a distinguished *output register* $r_o \in \mathcal{R}$;
- a labelling of transitions in $\mathcal{B}$ by copyless register updates.

▶ **Example 3** (Running example). The squaring function is recognised by a functional streaming string transducer which has two registers 1,2, with the output register being 1. The following picture shows this underlying automaton and its labelling by register updates.

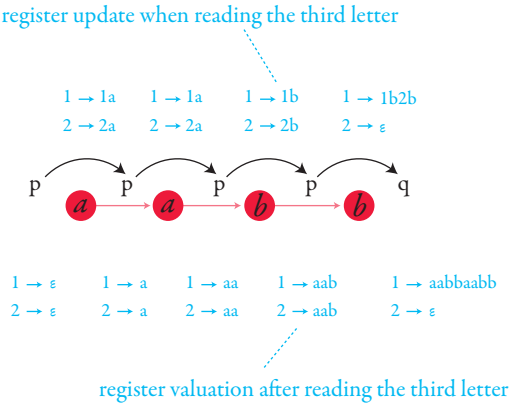


[1] Our syntax for streaming string transducers is different than the one used in [1], but it is routine to show that the expressive power is the same, as long as we allow nondeterminism.

The automaton uses nondeterminism to guess the last position so that the two registers are concatenated, however every nonempty input word admits exactly one run.

The semantics of a streaming string transducer is defined as follows. Suppose that ρ is a run of the underlying automaton $\mathcal{B}$, i.e., a sequence of transitions. Consider the empty register valuation which maps all registers to the empty word, and then apply all the register updates that label the transitions in ρ, beginning with the first transition and ending with the last transition. The output of ρ is defined to be the word over the output alphabet contained in the output register in the valuation described this way. Finally, the semantics of a streaming string transducer is defined to be the string-to-string transduction which consists of pairs (v, w) such that v is a word over the input alphabet and w is the output of some accepting run over the input word.

▶ **Example 4** (Running example)**.** Here is a picture of a run of the transducer from Example 3:



A streaming string transducer is called *unambiguous* if the automaton $\mathcal{B}$ admits at most one accepting run on every input word[2]. For an unambiguous streaming string transducer, its semantics is a partial function $\Sigma^* \to \Gamma^*$. The transducer in Example 3 is unambiguous.

2.2 Origin semantics and origin graphs

We now turn to the origin semantics of transducers. The idea is to give not just the output word, but also say which input positions were used to produce which output positions. The formalisation we use in this paper is *origin graphs*. Let fix an input alphabet Σ and an output alphabet Γ. For $w \in \Sigma^*$ and $v \in \Gamma^*$, an *origin graph* with input w and output v is defined to be a relational structure of the following form:

- the universe is the disjoint union of positions in w and positions in v;
- there are two binary predicates for the successor relations in w and v;
- there is a binary predicate, called the *origin mapping*, which is a total function from output positions to input positions;
- for each $a \in \Sigma \cup \Gamma$ there is a unary predicate which identifies positions with label a.

[2] One could also consider a streaming string transducer where the underlying automaton is deterministic. In this case, we would need to slightly modify the semantics, by adding a final function which performs a register update after reading the last input letter (e.g. concatenating the two registers as in the running example). After adding such final function, the deterministic model would have the same expressive power as the unambiguous variant used in this paper, see [1].

Note that the vocabulary of the relational structure depends on the choice of input and output alphabets; therefore a more formal definition would require talking about origin graphs over (Σ, Γ) where Σ is the input alphabet and Γ is the output alphabet. An origin graph can also be viewed as a directed graph, with vertices coloured by letters of the input and output alphabets, and edges coloured by three possible colours: successor edge in the input word, successor edge in the output word, and origin edge. Not every directed graph coloured this way is an origin graph; in an origin graph the input successors form a path, the output successors form a path on the remaining vertices, and the origin edges give a total function from the second path to the first one.

▶ **Definition 5.** An *origin string-to-string transduction* (origin transduction for short) consists of an input alphabet, an output alphabet, and a set of origin graphs over these alphabets.

Note that an origin transduction might contain origin graphs which differ only on the origin mapping. Here is an example picture, for the (not necessarily connected) subword relation equipped with the natural origin semantics:

An origin transduction is called *functional* if every input word appears in at most one origin graph. An example of a functional origin transduction is the squaring in our running example, when equipped with the natural origin information. A non-example is the subword relation.

Let us define the origin semantics of a streaming string transducer. When reading an input position x, the transducer executes a register update. Such a register update creates some new letters which are added to registers, and also moves the contents between registers. We assume that the origin of these created letters is the input position x, and remains this way even if the position is moved to different registers in subsequent transitions. Using this description, we can associate an origin graph to each run of the transducer. We say that an origin transduction is the *origin semantics of* (or to use an alternative name, *recognised by*) a streaming string transducer if it is the set of origin graphs corresponding to its successful runs. Note that, when the automaton is nondeterministic, different successful runs over the same input word and producing the same output word might generate different origin graphs.

3 MSO on origin graphs

In this section, we discuss properties of origin graphs that can be defined in monadic second-order logic MSO. This is the logic which extends first-order logic by allowing quantification over sets of elements in the universe (but not sets of pairs, nor sets of sets, etc.). For a definition of the syntax and semantics of MSO, see [9].

MSO on origin graphs. An origin graph is a special case of a relational structure. If the input and output alphabets are Σ and Γ, then the vocabulary of the relational structure consists of three binary relations (input edge, output edge, origin edge) as well as one unary predicate for each letter in $\Sigma \cup \Gamma$. We use the name *origin vocabulary of* (Σ, Γ) for this vocabulary. An MSO formula over this vocabulary defines a set of origin graphs, namely those

origin graphs where it is true. Note that the structures over the origin vocabulary which are origin graphs are a set definable in MSO, essentially because one can axiomatise in MSO (but not, e.g. in first-order logic) that a directed graph is a single finite directed path. Therefore when talking about MSO-definable sets of origin graphs, it makes no difference whether or not we require the MSO formula to check if a structure is actually an origin graph.

The following result shows that satisfiability of MSO over origin graphs produced by a given streaming string transducer is decidable.

▶ **Theorem 6.** *The following problem is decidable:*
Input: *A nondeterministic streaming string transducer $\mathcal{A}$ and an* MSO *formula φ over the origin vocabulary corresponding to $\mathcal{A}$;*
Question: *Is φ true in some origin graph in the origin semantics of $\mathcal{A}$?*

Proof Sketch. To prove this result it is convenient to use MSO transductions in the sense of Courcelle and Engelfriet, see [9]. We first convert $\mathcal{A}$ into a nondeterministic MSO transduction, which can be done while preserving origin semantics [5][3]. Given an MSO representation of $\mathcal{A}$, we can easily get an MSO transduction which inputs a word over the input alphabet, and outputs (non-deterministically) an origin graph that corresponds to some possible output of $\mathcal{A}$. Consider the following language

$$L = \{w \in \Sigma^* : \varphi \text{ is true in an origin graph produced by } \mathcal{A} \text{ on } w\}$$

By the Backward Translation Theorem (e.g. [9], p.66) the language L is definable in MSO, as the inverse image of an MSO-definable property under an MSO transduction. Therefore, L is regular, since MSO defines only regular word languages. ◀

▶ **Example 7.** An origin graph is called *order preserving* if the origin mapping gives a non-decreasing function from output positions to input positions. The set of order preserving origin graphs is clearly definable in MSO. Theorem 3 in [5] says that if an origin transduction has only order preserving origin graphs and is recognised by a streaming string transducer, then it is already recognised by a nondeterministic one-way transducer. Therefore Theorem 6 gives an algorithm for deciding if a streaming string transducer is equivalent, in terms of origin semantics, to a one-way nondeterministic transducer (such decidability, and even a polynomial time algorithm, although with inputs represented in a different way, was already given in [5]).

▶ **Example 8** (Running example)**.** Consider the squaring transduction. Every origin graph in this transduction satisfies the following property, which can be formalised in MSO: the output positions can be partitioned into two connected blocks, such that the origin mapping is order preserving when restricted to each of the blocks. For functional origin transductions, this property corresponds to being recognised by a deterministic two-way transducer which does two left-to-right passes on the input.

A corollary of the proof of Theorem 6 is that when the transducer $\mathcal{A}$ is fixed, then there is a linear time algorithm (which simply runs a finite automaton) for checking if a given input word can produce an origin graph satisfying φ.

▶ **Proposition 9.** *If an origin transduction is recognised by a streaming string transducer, then it is definable in* MSO *as a set of origin graphs.*

[3] In [5] the conversion is done in the deterministic case, but it can be easily extended to the nondeterministic case.

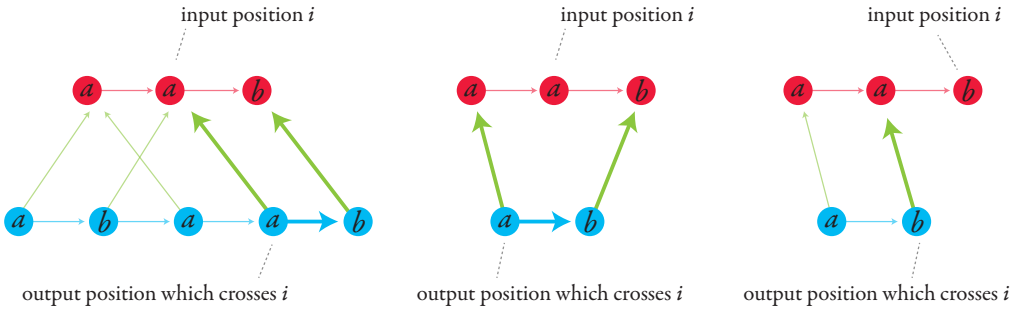
Note that the converse of the above proposition is false, even assuming that there is a bound on the number output positions which can originate in the same input position, see Examples 13 and 15. The issue is that it is more difficult to produce an origin graph (using the origin semantics of an MSO transduction) than it is to check if a given origin graph is correct.

4 Which sets of origin graphs are recognised by SSTs?

Which sets of origin graphs are recognised by streaming string transducers (equivalently, MSO transductions)?

What about functional streaming string transducers? The main goal of this paper is to give machine independent characterisations of such sets. This is Theorem 10 below, which says that a set of origin graphs is recognised by a streaming string transducer if and only if it is MSO-definable, has bounded origin (i.e., each input position is the origin of a bounded number of output positions), and it has bounded crossing, as explained below.

An output position j in the graph is said to *cross* an input position i if the position j has origin at most i and the successor of j either does not exist (i.e. j is the last output position) or has origin greater than i. Intuitively, to go from the origin of position j to the origin position $j+1$ on the input word, a reading head needs to cross position i. Here is a picture:



▶ **Theorem 10.** *Let $\mathcal{G}$ be an origin transduction, i.e., an input alphabet, an output alphabet, and a set of origin graphs over these alphabets. Then $\mathcal{G}$ is recognised by a streaming string transducer with k registers if and only if it satisfies all of the following conditions:*

bounded origin: *there is some $m \in \mathbb{N}$ such that in every origin graph from $\mathcal{G}$, every input position is the origin of at most m output positions;*

k-crossing: *in every origin graph from $\mathcal{G}$, every input position is crossed by at most k output positions;*

mso-definable: *there is an MSO formula which is true in exactly the origin graphs from $\mathcal{G}$.*

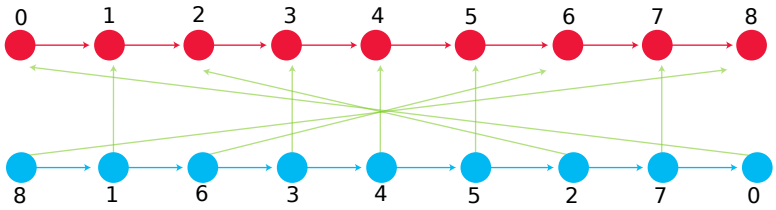
The theorem allows one to decide if a given SST can be implemented with fewer registers preserving origin semantics. This does not help with the resource minimisation problem from [4], because in [4] origin information can be be changed in the minimisation process.

The proof of the theorem is sketched in the next section. A corollary of the theorem is that an origin transduction is recognised by a streaming string transducer if and only if it has bounded origin, is MSO definable, and has bounded crossing (i.e., k-crossing for some k). The following three examples show how the three conditions in Theorem 10 are minimal, i.e., none of the conditions is implied by the remaining ones.

▶ **Example 11** (MSO definable)**.** Consider the identity origin transduction with its domain restricted to some non-regular subset of inputs, e.g. words of prime length. This origin transduction satisfies all conditions in Theorem 10 except for MSO definability.

▶ **Example 12** (Bounded origin)**.** Consider an origin transduction (with one letter in both the input and output alphabets) which is only defined on inputs with one letter and then copies the unique output letter an arbitrary number of times (therefore every output position originates in the unique input position). This origin transduction satisfies all conditions in Theorem 10 except for bounded origin. Streaming string transducers with ε-transitions, as discussed in Section 5, will be able to recognise this example.

▶ **Example 13** (Bounded crossing)**.** Consider the following origin transduction, which is functional. The input and output alphabets have only one letter each. The domain is words of odd length. A word of length $2n+1$ is mapped to a word of same length, but the origins are shuffled so that the origins of odd numbered positions are the same position, while the origins for even numbered positions are reversed. More precisely, if the positions are $0, \ldots, 2n$ then the origin of an odd numbered position $2i+1$ is $2i+1$, while the origin of an even numbered position $2i$ is $2n-2i$. Here is the picture of an origin graph in this origin transduction:



We claim that this origin transduction has unbounded crossing, but satisfies the remaining conditions in Theorem 10. To show unbounded crossing, observe that if the length of the input word is $2n+1$, then the middle input position n is crossed by all even numbered output positions greater than n. Clearly every origin graph in the transduction has bounded origin, because each input position is the origin of exactly one output position. For MSO definability, we observe that an origin graph belongs to the transduction if and only if it satisfies all the following conditions which are definable in MSO (in fact, first-order logic):

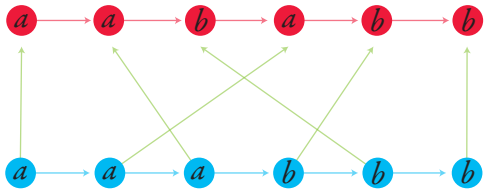
1. the origin of the first output position is the last input position;
2. the origin of the second output position is the second input position;
3. if $j > 1$ is an odd output position with origin i, then the origin of $j-2$ is $i-2$;
4. if $j > 1$ is an even output position with origin i, then the origin of $j-2$ is $i+2$.

Tree width. In Theorem 10, we use bounded crossing as one of the conditions. Another candidate for a structural property on origin graphs is that they have bounded tree width (see e.g. [9] for a definition). The following result shows that bounded tree width (and even bounded path width, which corresponds to the width of path decompositions, i.e., the special case of tree decompositions where the tree is a path) is a necessary condition for being recognised by a streaming string transducer. Indeed, we show that any origin transduction which is bounded crossing has bounded path width. Moreover, we can express the crossing boundedness property in terms of a particular path decomposition of bounded width.

▶ **Proposition 14.** *Every bounded crossing origin transduction has bounded path width.*

We now show that bounded path width is not a sufficient condition, in the sense that the bottom-up implication of Theorem 10 would fail if we would replace bounded crossing by bounded path width. One example is the origin transduction from Example 13, which can be shown to have bounded path width. Here is another example, which also shows that bounded crossing and recognisability by streaming string transducers are both notions that are not closed under reversing origin edges.

▶ **Example 15.** Consider a variant of the squaring function, which is defined only on words of even length, and maps a word w to uv where u (resp. v) is the subword consisting of the odd-numbered (resp. even-numbered) positions of w. This transduction is easily seen to be recognised by a (deterministic) streaming string transducer, and hence the underlying set $\mathcal{G}$ of origin graphs has bounded tree width by Proposition 14. Define $\mathcal{G}'$ to be the set of origin graphs which are obtained from $\mathcal{G}$ by reversing the origin edges. Here is a picture of an origin graph in $\mathcal{G}'$:



Since the origin mapping is bijective, $\mathcal{G}'$ is also a set of origin graphs. It has bounded origin (bounded by 1). By Theorem 10 and Proposition 14, $\mathcal{G}$ is MSO definable and has bounded path width. Path width and MSO definability are not changed by reversing arrows, and therefore $\mathcal{G}'$ has bounded origin, bounded path width and is also MSO-definable. Nevertheless, $\mathcal{G}'$ is not the origin semantics of any streaming string transducer. Indeed, if $\mathcal{A}$ would be a streaming string transducer recognising $\mathcal{G}'$, then the pre-image under $\mathcal{A}$ of the regular set $(aa + bb)^*$ would be the non-regular set of words of the form ww over the alphabet $\{a, b\}$, contradicting the fact that regular word languages are preserved under taking pre-images of streaming string transducers (essentially the Backwards Translation Theorem from [9]).

Recognisability. In Theorem 10, the conditions used are bounded origin, bounded crossing and MSO definability. While bounded origin and bounded crossing are purely combinatorial properties of graphs, MSO definability has a more syntactic character. A less syntactic alternative to MSO definability would be to use *recognisability* in the sense of Definition 4.29 in [8]. Intuitively speaking, a class of relational structures is called recognisable if it has finite index for a certain naturally defined equivalence relation à la Myhill Nerode. In [6] it is shown that if a class of relational structures has bounded tree width, then recognisability is the same as thing as definability in MSO. Therefore, in the statement of Theorem 10 we could replace MSO definability by recognisability, and the theorem would still be true.

The functional case. Theorem 10 gives a characterisation of origin transductions recognised by streaming string transducers. Recall that a streaming string transducer was called unambiguous if its underlying automaton had at most one successful run for each input word. Such transducers are equivalent to the deterministic model in [3], also when using origin semantics [5]. They can only recognise functional origin transductions. As it turns out, this is the only restriction, i.e., if an origin transduction is functional and recognised by a (possibly ambiguous) streaming string transducer, then it is recognised by an unambiguous one. Furthermore, in the functional case the condition on bounded origin becomes superfluous.

▶ **Theorem 16.** *Let $\mathcal{G}$ be an origin transduction. Then $\mathcal{G}$ is recognised by an unambiguous streaming string transducer with k registers if and only if it is functional, k-crossing and* MSO*-definable.*

Unambiguous streaming string transducers can moreover be simulated by deterministic ones (in the sense of Alur and Černý [1]) but at the cost of adding registers [2].

5 Sketch of the proof

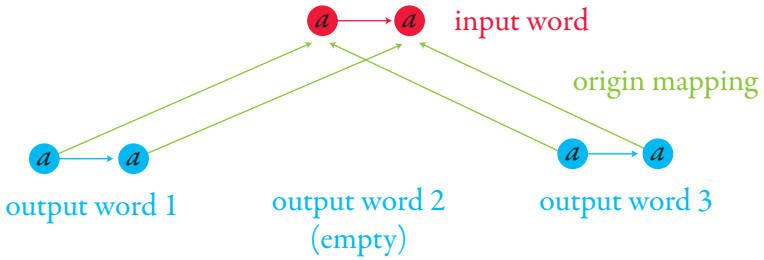
To prove Theorems 10 and 16, we first characterise the origin graphs which can be generated by streaming string transducers with ε-transitions. A streaming string transducer with ε-transitions is the generalisation of the model described in Definition 2, where ε-transitions are allowed in the underlying automaton. The origin mapping is defined so that if an output position is created (i.e., added to some register) by an ε-transition, then its origin is the most recently read input letter. To make this well defined, we make the syntactic restriction that no ε-transitions can be used when in an initial state, and therefore the first transition in each run must consume an input letter.

▶ **Example 17.** Consider the set of origin graphs where the input word has only one letter, and hence this letter is the origin of all output positions, and the output word is $a^n b^n$ for some $n \in \mathbb{N}$. To recognise this string transduction, we use two registers. After reading the unique input position, the automaton enters a loop of ε-transitions. Each one appends a to one register, and b to the other. At the end, the transducer does an ε-transition to the accepting state which concatenates both registers.

The above example shows that when ε-transitions are allowed, the origin semantics of a streaming string transducer needs no longer to be MSO-definable. Theorem 18 below shows that if we additionally assume that a set of origin graphs is MSO-definable, then being the semantics of a streaming string transducer with ε-transitions is equivalent to having bounded crossing. The theorem is the main step in our proof of Theorems 10 and 16.

▶ **Theorem 18.** *Let $\mathcal{G}$ be an origin transduction which is* MSO*-definable. Then, $\mathcal{G}$ is recognised by a k-register streaming string transducer with ε-transitions if and only if $\mathcal{G}$ is k-crossing.*

We sketch the proof here. The left-to-right implication is straightforward, and does not need the assumption on MSO definability. Below we discuss the converse implication. The idea is that if an origin graph has bounded crossing, then it can be constructed by applying a sequence of elementary operations to the empty graph. Intermediate objects produced by these elementary operations are going to be like origin graphs, except that the output word might be in several pieces, corresponding intuitively to the register contents. Define a *k-block origin graph* to be the extension of origin graphs where there are exactly k output words, which are called *blocks*, some of which may be empty, as in the following picture for $k = 3$:



To transform k-block origin graphs, we use the following toolkit of operations, which corresponds intuitively to the registers in a streaming string transducer.

Input. This operation takes one parameter, a letter a of the input alphabet. The result of the operation is that a new position with letter a is added to the end of the input word;

Output. This operation takes three parameters: a *target block* $i \in \{1, \ldots, k\}$, a *content* c which is either a letter of the output alphabet or a number $j \in \{1, \ldots, k\}$ different than i,

and a *side* $s \in \{\text{left,right}\}$. The result of the operation is that the content (i.e., either an output letter of or the contents of register c, depending on the type of c) is concatenated to the left/right (depending on the side s) of the target block i. If the content is a letter, then its origin is set to be the last input position (if there is no input position and the content is a letter, then the operation fails).

We write Ω_k for the above set of operations (assuming that the alphabets are implicit from the context), which is finite. Define *k-folding* to be the function from $(\Omega_k)^*$ to k-block origin graphs which maps a sequence of operations to the k-block origin graph obtained by successively applying the sequence of operations starting from the empty k-block origin graph. Note that k-folding is partial, because the output operation can fail.

▶ **Lemma 19.** *The k-folding operation is (a) surjective; and (b) an* MSO *interpretation.*

Proof of Theorem 18. We only show the right-to-left implication. Suppose then that $\mathcal{G}$ is an origin transduction which is MSO definable and is k-crossing. Define $\mathcal{G}'$ to be the set of k-block origin graphs such that: (i) the i-th output word is empty for $i \neq 1$; and (ii) if only the input and 1-st output word are kept, the resulting origin graph belongs to $\mathcal{G}$. If $\mathcal{G}$ is MSO definable, then so is $\mathcal{G}'$. Let $L \subseteq (\Omega_k)^*$ be those sequences of operations whose k-folding is in $\mathcal{G}'$. By Lemma 19 (b) and the Backwards Translation Theorem [9], L is definable in MSO. By Lemma 19 (a), $\mathcal{G}'$ is equal to the image of L under k-folding. By Büchi-Elgot-Trakhtenbrot's Theorem [7], L is recognised by a finite automaton. We transform this finite automaton into a k-register nondeterministic streaming string transducer with ε-transitions by translating any letter σ of Ω_k into:

- a transition reading an input symbol without updating the registers, if σ is of type input;
- an ε-transition with an appropriate register operation if σ is of type output. ◀

To complete the proof of Theorem 10, we finally show that if the origin semantics of an εNSST has bounded origin then ε-transitions can be eliminated.

6 Classes of origin transductions and perspectives

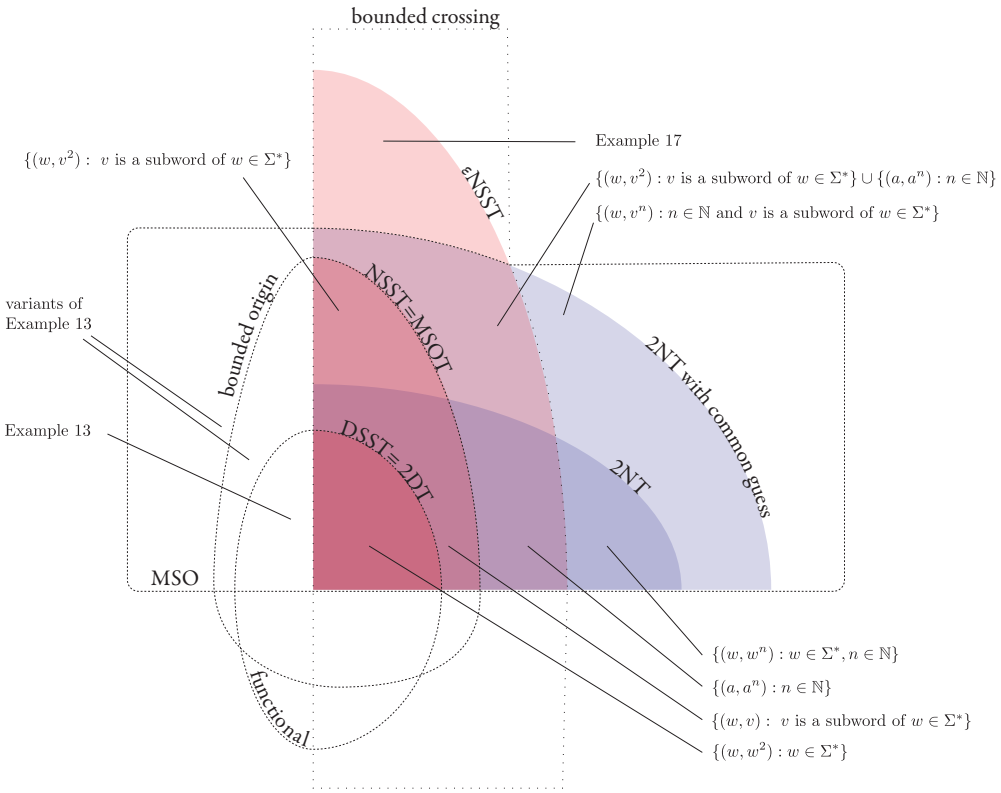
Our main contribution is a characterisation of the origin semantics of streaming string transducers (deterministic; nondeterministic; with ε-transitions providing MSO-definability), using properties of the origin graphs such as functionality, origin boundedness, crossing boundedness and MSO-definability. The origin transductions recognised by these transducers form a hierarchy depicted in red in the figure below, where DSST (resp. NSST, εNSST) denotes the family of origin transductions recognised by a deterministic (resp. nondeterministic, nondeterministic with ε-transitions) streaming string transducer.

The figure also includes two-way transducers, which define an orthogonal hierarchy, depicted in blue. We consider deterministic and nondeterministic variants as well as those with *common guess*. A two-way transducer (deterministic or not) is equipped with a common guess if, before starting the computation, a finite colouring of the input positions is performed, and this colouring is the same each time the head revisits a position. This strictly increases the expressivity, e.g. the relation $\{(u, vv) \mid v \text{ is a subword of } u\}$ is recognised by a two-way transducer with common guess but not without. The origin semantics of a two-way transducer is defined in a natural way, i.e., an output letter originates in the input position which was scanned by the input head when the letter has been produced. In the figure, the classes of origin transductions recognised by two-way automata are denoted respectively by 2DT, 2NT, and 2NT with common guess.

The class of functions recognised by deterministic two-way transducers (resp. with common guess) is known to be the same as the one recognised by deterministic streaming string transducers [11, 1] (resp. nondeterministic streaming string transducers [3]), even when considering the origin semantics [5].

Finally, we denote by MSOT the family of origin transductions recognised by a nondeterministic MSO-transduction. It is equal to NSST [3, 11]. The transductions recognised by deterministic MSO-transductions are the same as for DSST [1], this remains true with origin semantics [5].

We can prove that all the MSO-definable (resp. bounded origin) origin transductions in εNSST are recognised by a nondeterministic two-way transducer with common guess (resp., are in NSST). Moreover, all the transductions that have bounded origin and which are recognised by a nondeterministic two-way transducer with common-guess, are recognised by a streaming string tranducer (and have therefore bounded crossing). All the other intersections are nonempty and can be populated with some origin transductions (with their natural origin information) as depicted in the figure.



References

1 Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, pages 1–12, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.1.

2 Rajeev Alur and Loris D'Antoni. Streaming tree transducers. In *International Colloquium on Automata, Languages, and Programming*, pages 42–53. Springer, 2012. URL: http://link.springer.com/chapter/10.1007/978-3-642-31585-5_8.

3 Rajeev Alur and Jyotirmoy V. Deshmukh. Nondeterministic streaming string transducers. In *International Colloquium on Automata, Languages, and Programming*, pages 1–20. Springer, 2011. URL: `http://link.springer.com/10.1007%2F978-3-642-22012-8_1`.

4 Félix Baschenis, Olivier Gauwin, Anca Muscholl, and Gabriele Puppis. Minimizing resources of sweeping and streaming string transducers. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.

5 Mikołaj Bojańczyk. Transducers with origin information. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 26–37, 2014. `doi:10.1007/978-3-662-43951-7_3`.

6 Mikołaj Bojańczyk and Michal Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'16, New York, NY, USA, July 5-8, 2016*, pages 407–416, 2016. `doi:10.1145/2933575.2934508`.

7 J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92, 1960. `doi:10.1002/malq.19600060105`.

8 Bruno Courcelle. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. *Theor. Comput. Sci.*, 80(2):153–202, 1991. `doi:10.1016/0304-3975(91)90387-H`.

9 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012. URL: `http://www.cambridge.org/fr/knowledge/isbn/item5758776/?site_locale=fr_FR`.

10 Luc Dartois, Emmanuel Filiot, and Nathan Lhote. Decidable logics for transductions and data words. *CoRR*, abs/1701.03670, 2017. URL: `http://arxiv.org/abs/1701.03670`.

11 Joost Engelfriet and Hendrik Jan Hoogeboom. Mso definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001. `doi:10.1145/371316.371512`.

Continuity and Rational Functions*

Michaël Cadilhac[1], Olivier Carton[2], and Charles Paperman[3]

1 **WSI, Universität Tübingen, Tübingen, Germany**
Michael@Cadilhac.name
2 **IRIF, Université Paris Diderot, Paris, France**
Olivier.Carton@irif.fr
3 **WSI, Universität Tübingen, Tübingen, Germany**
Charles.Paperman@gmail.com

Abstract

A word-to-word function is continuous for a class of languages $\mathcal{V}$ if its inverse maps $\mathcal{V}$_languages to $\mathcal{V}$. This notion provides a basis for an algebraic study of transducers, and was integral to the characterization of the sequential transducers computable in some circuit complexity classes.

Here, we report on the decidability of continuity for functional transducers and some standard classes of regular languages. Previous algebraic studies of transducers have focused on the structure of the underlying input automaton, disregarding the output. We propose a comparison of the two algebraic approaches through two questions: When are the automaton structure and the continuity properties related, and when does continuity propagate to superclasses?

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Transducers, rational functions, language varieties, continuity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.115

1 Introduction

The algebraic theory of regular languages is tightly interwoven with fundamental questions about the computing power of Boolean circuits and logics. The most famous of these braids revolves around $\mathcal{A}$, the class of *aperiodic* or *counter-free* languages. Not only is it expressed using the logic FO[<], but it can be seen as the basic building block of AC^0, the class of languages recognized by circuit families of polynomial size and constant depth, this class being in turn expressed by the logic FO[arb] (see [18] for a lovely account). This pervasive interaction naturally prompts to lift this study to the functional level, hence to *rational functions*. This was started in [4], where it was shown that a subsequential (i.e., input-deterministic) transducer computes an AC^0 function iff it preserves the regular languages of AC^0 by inverse image. Buoyed by this clean, semantic characterization, we wish to further investigate this latter property for different classes: say that a function $f\colon A^* \to B^*$ is $\mathcal{V}$_continuous, for a class of languages $\mathcal{V}$, if for every language $L \subseteq B^*$ of $\mathcal{V}$, the language $f^{-1}(L)$ is also a language of $\mathcal{V}$. Our main focus will be on deciding $\mathcal{V}$_continuity for rational functions; before listing our main results, we emphasize two additional motivations.

First, there has been some historical progression towards this goal. Noting, in [9], that inverse rational functions provide a uniform and compelling view of a wealth of natural operations on regular languages, Pin and Sakarovitch initiated in [10] a study of regular-continuous functions. It was already known at the time, by a result of Choffrut (see [3,

* The first and third authors are partly funded by the DFG Emmy Noether program (KR 4042/2); the second author is funded by the DeLTA project (ANR-16-CE40-0007).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 115; pp. 115:1–115:14

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

Theorem 2.7]), that regular-continuity together with some uniform continuity property *characterize* functions computed by subsequential transducers. This characterization was instrumental in the study of Reutenauer and Schützenberger [15], who already noticed the peculiar link between uniform continuity for some distances on words and continuity for certain classes of languages. This link was tightened by Pin and Silva [11] who formalized this topological approach and generalized it to rational relations. More recently [12], the same authors made precise the link unveiled by Reutenauer and Schützenberger, and developed a fascinating and robust framework in which language continuity has a topological interpretation (see the beginning of Section 3, as we build upon this theory). Pin and Silva [13] notably proposed thereafter a study of functions that propagate continuity for a class to subclasses.

Second, the interweaving between languages, circuits, and logic that was alluded to previously can in fact be formally stated (see again [18, 19]). As a central property towards this formalization is the correspondence between "cascade products" of automata, stacking of circuits, and nesting of formulas, respectively. Strikingly, these operations can all be seen as inverse rational functions [19]. These operations being intrinsic in the construction of complex objects, decompositions are often naturally used to specify languages, circuits, and formulas (see, e.g., [17, Section 5.5]). We remark that a sufficient condition for the result of the composition to be in some given class (of languages, circuits, or logic formulas), is that each rational function be continuous for that class. Hence deciding continuity allows to give a sufficient condition for this membership question *without* computing the result of the composition, which is subject to combinatorial blowup.

Here, we report on three questions, the first two relating continuity to the main other algebraic approach to transducers, while allowing a more gentle introduction to the evaluation of *profinite words* by transducers:

- When is the transducer *structure* (i.e., its so-called *transition monoid*) impacting its continuity? The results of Reutenauer and Schützenberger [15] can indeed be seen as the starting point of two distinct algebraic theories for rational functions; on the one hand, the study of continuity, and on the other the study of the transition monoid of the transducer (by disregarding the output). This latter endeavor was carried by [5].
- What is the impact of *variety inclusion* on the inclusion of the related classes of continuous rational functions? When the focus is solely on the structure of the transducer, there is a natural propagation to superclasses; when is it the case for continuity?
- When is $\mathcal{V}$_continuity decidable for rational functions? We show decidability for the varieties $\mathcal{J}, \mathcal{R}, \mathcal{L}, \mathcal{DA}, \mathcal{A}, \mathcal{C}om, \mathcal{A}b, \mathcal{G}_{\mathrm{sol}}$, and $\mathcal{G}$; these constitute our main results.

2 Preliminaries

We assume some familiarity with the theory of automata and transducers, and concepts related to metric spaces (see, e.g., [3, 8] for presentations pertaining to our topic). Apart from these prerequisites, for which the notation is first settled, the presentation is self-contained.

We will use A and B for alphabets, and A^* for words over A, with 1 the empty word. For each word u, there is a smallest v, called the *primitive root* of u, such that $u = v^c$ for some c; if $c = 1$, then u is itself *primitive*. We write $|u|$ for the length of a word $u \in A^*$ and $\mathsf{alph}(u)$ for the set of letters that appear in u. For a word $u \in A^*$ and a language $L \subseteq A^*$, we write $u^{-1}L$ for $\{v \mid u \cdot v \in L\}$, and symmetrically for Lu^{-1}, these two operations being called the left and right quotients of L by u, respectively. We naturally extend concatenation and quotients to relations, in a component-wise fashion, e.g., for $R \subseteq A^* \times A^*$ and a pair $\rho \in A^* \times A^*$, we may use $\rho^{-1}R$ and $R\rho^{-1}$. We write L^{c} for the complement of L. A *variety* is

a mapping $\mathcal{V}$ which associates with each alphabet A a set $\mathcal{V}(A^*)$ of regular languages closed under the Boolean operations and quotient, and such that for any morphism $h\colon A^* \to B^*$ and any $L \in \mathcal{V}(B^*)$, it holds that $h^{-1}(L) \in \mathcal{V}(A^*)$. Reg is the variety that maps every alphabet A to the set $\mathrm{Reg}(A^*)$ of regular languages over A. Given two languages $K, L \subseteq A^*$, we say that they are $\mathcal{V}$_separable if there is a $S \in \mathcal{V}(A^*)$ such that $K \subseteq S$ and $L \cap S = \emptyset$.

Transducers. A transducer τ is a 9-tuple $(Q, A, B, \delta, I, F, \lambda, \mu, \rho)$ where (Q, A, δ, I, F) forms an automaton (i.e., Q is a state set, A an input alphabet, $\delta \subseteq Q \times A \times Q$ a transition set, $I \subseteq Q$ a set of initial states, and $F \subseteq Q$ a set of final states), and additionally, B is an output alphabet and $\lambda\colon I \to B^*, \mu\colon \delta \to B^*, \rho\colon F \to B^*$ are the output functions. We write $\tau_{q,q'}$ for τ with $I := \{q\}$ and $F := \{q'\}$, adjusting λ and ρ to output 1 if they were undefined on these states. Similarly, $\tau_{q,\bullet}$ is τ with $I := \{q\}$ and F unchanged, and symmetrically for $\tau_{\bullet,q}$. For $q \in Q$ and $u \in A^*$, we write $q.u$ for the set of states reached from q by reading u. We assume that all the transducers and automata under study have no useless state, that is, that all states appear in some accepting path.

With $w \in A^*$, let $t\_1 t\_2 \cdots t\_|w| \in \delta^*$ be an accepting path for w, starting in a state $q \in I$ and ending in some $q' \in F$. The output of this path is $\lambda(q)\mu(t\_1)\mu(t\_2)\cdots\mu(t\_n)\rho(q')$, and we write $\tau(w)$ for the set of outputs of such paths. We use τ for both the transducer and its associated partial function from A^* to subsets of B^*. Relations of the form $\{(u, v) \mid v \in \tau(u)\}$ are called *rational relations*.

The transducer τ is *unambiguous* if there is at most one accepting path for each word. In that case $\tau_{q,q'}$ is also an unambiguous transducer for any states q, q'. When τ is unambiguous, it realizes a word-to-word function: the set of functions computed by unambiguous transducers is the set of *rational functions*. Further restricting, if the underlying automaton is deterministic, we say that τ is *subsequential*. If τ is a finite union of subsequential rational functions of disjoint domains, we say that τ is *plurisubsequential*.

Word distances, profinite words. For a variety $\mathcal{V}$ of regular languages, we define a distance between words for which, intuitively, two words are close if it is hard to separate them with $\mathcal{V}$ languages. Define $d\_\mathcal{V}(u, v)$, for words $u, v \in A^*$, to be 2^{-r} where r is the size of the smallest automaton that recognizes a language of $\mathcal{V}(A^*)$ that separates $\{u\}$ from $\{v\}$; if no such language exists, then $d\_\mathcal{V}(u, v) = 0$. It can be shown that this distance is a *pseudo-ultrametric* [8, Section VII.2]; we make only implicit and innocuous use of this fact.

We simply write d for d_Reg. The complete metric space that is the completion of (A^*, d) is denoted $\widehat{A^*}$ and is called the *free profinite monoid*, its elements being the *profinite words*, and the concatenation being naturally extended. By definition, if $(u\_n)\_n > 0$ is a Cauchy sequence, it should hold that for any regular language L, there is a N such that either all u_n with $n > N$ belong to L, or none does. For any $x \in A^*$, define the profinite word $x^\omega = \lim x^{n!}$, and more generally, $x^{\omega - c} = \lim x^{n! - c}$. That $(x^{n!})\_n > 0$ is a Cauchy sequence is a starting point of the profinite theory [8, Proposition VI.2.10]; it is also easily checked that $x^{c \times \omega} = \lim x^{c \times n!}$ is equal to x^ω for any integer $c \geq 1$. Given a language $L \subseteq A^*$, we write $\overline{L} \subseteq \widehat{A^*}$ for its closure, and we note that if L is regular, $\overline{L}^{\mathrm{c}} = \overline{L^{\mathrm{c}}}$ and for L' regular, $\overline{L \cup L'} = \overline{L} \cup \overline{L'}$, and similarly for intersection (see [8, Theorem VI.3.15]).

Equations. For $u, v \in \widehat{A^*}$, a language $L \subseteq A^*$ *satisfies the (profinite) equation* $u = v$ if for any words $s, t \in A^*$, $[s \cdot u \cdot t \in \overline{L} \Leftrightarrow s \cdot v \cdot t \in \overline{L}]$. Similarly, a class of languages satisfies an equation if all the languages of the class satisfy it. For a variety $\mathcal{V}$, we write $u = \_\mathcal{V} v$, and

say that u is equal to v in $\mathcal{V}$, if $\mathcal{V}(A^*)$ satisfies $u = v$. For a partial function f, $f(u) = \_\mathcal{V} f(v)$ means that either both $f(u)$ and $f(v)$ are undefined, or they are both defined and equal in $\mathcal{V}$.

Given a set E of equations over $\widehat{A^*}$, the class of languages *defined* by E is the class of languages over A^* that satisfy all the equations of E. Reiterman's theorem shows in particular that for any variety $\mathcal{V}$ and any alphabet A, $\mathcal{V}(A^*)$ is defined by a set of equations (the precise form of which being studied in [6]).

More on varieties. Borrowing from Almeida and Costa [2], we say that a variety $\mathcal{V}$ is *supercancellative* when for any alphabet A, any $u, v \in \widehat{A^*}$ and $x, y \in A$, if $u \cdot x = \_\mathcal{V} v \cdot y$ or $x \cdot u = \_\mathcal{V} y \cdot v$, then $u = \_\mathcal{V} v$ and $x = y$. This implies in particular that for any word $w \in A^*$, both $w \cdot A^*$ and $A^* \cdot w$ are in $\mathcal{V}(A^*)$. We further say that a variety $\mathcal{V}$ *separates words* if for any $s, t \in A^*$, $\{s\}$ and $\{t\}$ are $\mathcal{V}$_separable.

Our main applications revolve around some classical varieties, that we define over any possible alphabet A as follows, where x, y range over all of A^*, and a, b over A:

- $\mathcal{J}$, def. by $(xy)^\omega \cdot x = y \cdot (xy)^\omega = (xy)^\omega$
- $\mathcal{R}$, def. by $(xy)^\omega \cdot x = (xy)^\omega$
- $\mathcal{L}$, def. by $y \cdot (xy)^\omega = (xy)^\omega$
- $\mathcal{DA}$, def. by $x^\omega \cdot z \cdot x^\omega = x^\omega$ for all $z \in \mathsf{alph}(x)^*$
- $\mathcal{A}$, def. by $x^{\omega+1} = x^\omega$
- $\mathcal{Com}$, def. by $ab = ba$
- $\mathcal{A}_\mathcal{B}$, def. by $ab = ba$ and $a^\omega = 1$
- $\mathcal{G}_{\text{nil}}$, the languages rec. by nilpotent groups
- $\mathcal{G}_{\text{sol}}$, the languages rec. by solvable groups
- $\mathcal{G}$, the languages rec. by groups

The varieties included in $\mathcal{A}$ are called *aperiodic varieties* and those in $\mathcal{G}$ are called *group varieties*. Precise definitions, in particular for the group varieties, can be found in [18, 14]; we simply note that in group varieties, x^ω equals 1 for all $x \in A^*$. All these varieties except for $\mathcal{A}_\mathcal{B}$ and $\mathcal{Com}$ separate words, and only $\mathcal{DA}$ and $\mathcal{A}$ are supercancellative. They verify:

$$\begin{array}{ccc} & \subsetneq\ \mathcal{R}\ \supsetneq & \\ \mathcal{J} = \mathcal{R} \cap \mathcal{L} & \mathcal{DA} \subsetneq \mathcal{A} & \\ & \supsetneq\ \mathcal{L}\ \subsetneq & \end{array} \qquad \begin{array}{c} \subsetneq\ \mathcal{Com} \\ \mathcal{A}_\mathcal{B} = \mathcal{G} \cap \mathcal{Com} \subsetneq \mathcal{G}_{\text{nil}} \subsetneq \mathcal{G}_{\text{sol}} \subsetneq \mathcal{G} \end{array}$$

On transducers and profinite words. For a profinite word u and a state q of an unambiguous transducer τ, the set $q.u$ is well-defined; indeed, with $u = \lim u\_n$, the set $q.u\_n$ is eventually constant, as otherwise for some state q', the domain of $\tau_{q,q'}$ would be a regular language that separates infinitely many u_n's.

A transducer $\tau\colon A^* \to B^*$ is a *$\mathcal{V}$_transducer*,[1] for a variety $\mathcal{V}$, if for some set of equations E defining $\mathcal{V}(A^*)$, for all $(u = v) \in E$ and all states q of τ, it holds that $q.u = q.v$. A rational function is *$\mathcal{V}$_realizable* if it is realizable by a $\mathcal{V}$_transducer.

Continuity. For a variety $\mathcal{V}$, a function $f\colon A^* \to B^*$ is $\mathcal{V}$_continuous[2] iff for any $L \in \mathcal{V}(B^*)$, $f^{-1}(L) \in \mathcal{V}(A^*)$. We mostly restrict our attention to rational functions, and their being

[1] The usual definition of $\mathcal{V}$_transducer is based on the so-called transition monoid of τ, see, e.g., [15]; the definition here is easily seen to be equivalent by [1, Lemma 3.2] and [4, Lemma 1].

[2] A note on terminology: There has been some fluctuation on the use of the term "continuous" in the literature, mostly when a possible incompatibility arises with topology. In [13], the authors use the term "preserving" in the more general context of functions from monoids to monoids. In our study, we focus on word to word functions, in which the natural topological context provides a solid basis for the use of "continuous," as used in [11, 4].

computed by transducers implies that they are countably many. We note that much more Reg_continuous functions exist, in particular uncomputable ones:

▶ **Proposition 1.** *There are uncountably many Reg_continuous functions.*

3 Continuity: The profinite approach

We build upon the work of Pin and Silva [11] and develop tools specialized to rational functions. In Section 3.1, we present a lemma asserting the equivalence between $\mathcal{V}$_continuity and the "preservation" of the defining equations for $\mathcal{V}$. In the sections thereafter, we specialize this approach to rational functions. As noted in [11], it often occurs that results about rational functions can be readily applied to the larger class of Reg_continuous functions; here, this is in particular the case for the Preservation Lemma of Section 3.1.

Our main appeal to a classical notion of continuity is given by the:

▶ **Theorem 2** ([12, Theorem 4.1])**.** *Let $f\colon A^* \to B^*$. It holds that f is $\mathcal{V}$_continuous iff f is uniformly continuous for the distance $d\_\mathcal{V}$.*

Consequently, if f is Reg_continuous then it has a unique extension to the free profinite monoids, written $\widehat{f}\colon \widehat{A^*} \to \widehat{B^*}$. The salient property of this mapping is that it is continuous in the *topological sense* (see, e.g., [8]). For our specific needs, we simply mention that it implies that for any regular language L, we have that $\widehat{f}^{-1}(\overline{L})$ is closed (that is, it is the closure of some set).

3.1 The Preservation Lemma: Continuity is preserving equations

The Preservation Lemma gives us a key characterization in our study: it ties together continuity and some notion of preservation of equations. This can be seen as a generalization to functions of equation satisfaction for languages. We will need the following technical lemma that extends [8, Proposition VI.3.17] from morphisms to arbitrary Reg_continuous functions; interestingly, this relies on a quite different proof.

▶ **Lemma 3.** *Let $f\colon A^* \to B^*$ be a Reg_continuous function and L a regular language. It holds that $\widehat{f}^{-1}(\overline{L}) = \overline{f^{-1}(L)}$.*

▶ **Lemma 4** (Preservation Lemma)**.** *Let $f\colon A^* \to B^*$ be a Reg_continuous function and E a set of equations that defines $\mathcal{V}(A^*)$. The function f is $\mathcal{V}$_continuous iff for all $(u = v) \in E$ and words $s, t \in A^*$, $\widehat{f}(s \cdot u \cdot t) = \_\mathcal{V}\widehat{f}(s \cdot v \cdot t)$.*

Proof. *(Only if)* Suppose f is $\mathcal{V}$_continuous. Let $u, v \in \widehat{A^*}$ such that $u = \_\mathcal{V}v$, and $s, t \in A^*$. Since by $\mathcal{V}$_continuity $f^{-1}(B^*) \in \mathcal{V}(A^*)$, either both $s \cdot u \cdot t$ and $s \cdot v \cdot t$ belong to the closure of this language, or they both do not. The latter case readily yields the result, hence suppose we are in the former case.

By definition, $u = \lim u\_n$ and $v = \lim v\_n$ for some Cauchy sequences of words $(u\_n)\_n > 0$ and $(v\_n)\_n > 0$. Since $s \cdot u \cdot t = \_\mathcal{V}s \cdot v \cdot t$, the hypothesis yields that $d\_\mathcal{V}(s \cdot u\_n \cdot t, s \cdot v\_n \cdot t)$ tends to 0. By Theorem 2, f is uniformly continuous for $d\_\mathcal{V}$, hence $d\_\mathcal{V}(f(s \cdot u\_n \cdot t), f(s \cdot v\_n \cdot t))$ also tends to 0 (note that both $f(s \cdot u\_n \cdot t)$ and $f(s \cdot v\_n \cdot t)$ are defined for all n big enough). This shows that $\widehat{f}(s \cdot u \cdot t) = \_\mathcal{V}\widehat{f}(s \cdot v \cdot t)$.

(If) Suppose that f preserves the equations of E as in the statement. Let $L \in \mathcal{V}(B^*)$, we wish to verify that $L' = f^{-1}(L) \in \mathcal{V}(A^*)$, or equivalently by definition, that L' satisfies all the equations of E. Let $(u = v) \in E$ be one such equation, and $s, t \in A^*$; we must show that $s \cdot u \cdot t \in \overline{L'} \Leftrightarrow s \cdot v \cdot t \in \overline{L'}$.

Suppose $s \cdot u \cdot t \in \overline{L'}$. Since f is Reg_continuous, it holds that $\widehat{f}(s \cdot u \cdot t) \in \overline{L}$ (observe that $\widehat{f}(s \cdot u \cdot t)$ is indeed defined). By hypothesis, $\widehat{f}(s \cdot u \cdot t) = \_\mathcal{V}\widehat{f}(s \cdot v \cdot t)$; now since $L \in \mathcal{V}(B^*)$, it must hold that $\widehat{f}(s \cdot v \cdot t) \in \overline{L}$. Taking the inverse image of $\widehat{f}$ on both sides, it thus holds that $s \cdot v \cdot t \in \widehat{f}^{-1}(\overline{L})$, and Lemma 3 then shows that $s \cdot v \cdot t \in \overline{L'}$. As the argument works both ways, this shows that $s \cdot u \cdot t \in \overline{L'} \Leftrightarrow s \cdot v \cdot t \in \overline{L'}$, concluding the proof. ◀

Continuity can be seen as preserving *membership* to $\mathcal{V}$ (by inverse image); this is where the nomenclature "$\mathcal{V}$_preserving function" of [13] stems from. Strikingly, this could also be worded as preserving *nonmembership* to $\mathcal{V}$:

▶ **Proposition 5.** *A Reg_continuous total*[3] *function $f\colon A^* \to B^*$ is $\mathcal{V}$_continuous iff for all $L \subseteq A^*$ that do* not *belong to $\mathcal{V}(A^*)$, $f(L)$ and $f(L^c)$ are not $\mathcal{V}$_separable.*

3.2 The profinite extension of rational functions

The Preservation Lemma already hints at our intention to see transducers as computing functions from and to the free profinite monoids. Naturally, if τ is a rational function, its being Reg_continuous allows us to do so (by Theorem 2). For $u = \lim u\_n$ a profinite word, we will write $\tau(u)$ for $\widehat{\tau}(u)$, i.e., the limit $\lim \tau(u\_n)$, which exists by continuity. In this section, we develop a slightly more combinatorial approach to this evaluation, and address two classes of profinite words: those expressed as $s \cdot u \cdot t$ for s, t words and u a profinite word, and those expressed as x^ω for x a word.

Recall that for a transducer state q and a profinite word u, $q.u$ is well-defined. As a consequence, if s and t are words and τ is unambiguous, then there is at most one initial state q_0, one $q \in q\_0.s$ and one $q' \in q.u$ such that $q'.t$ is final, and these states exist iff $\tau(s \cdot u \cdot f)$ is defined. Thus:

▶ **Lemma 6.** *Let τ be an unambiguous transducer from A^* to B^*, $s, t \in A^*$ and $u \in \widehat{A^*}$. Suppose $\tau(s \cdot u \cdot f)$ is defined, and let q_0, q, q' be the unique states such that q_0 is initial, $q \in q\_0.s$, $q' \in q.u$, and $q'.t$ is final. The following holds: $\tau(s \cdot u \cdot t) = \tau_{\bullet,q}(s) \cdot \tau_{q,q'}(u) \cdot \tau_{q',\bullet}(t)$.*

▶ **Lemma 7.** *Let τ be an unambiguous transducer from A^* to B^* and $x \in A^*$. If $\tau(x^\omega)$ is defined, then there are words $s, y, t \in B^*$ such that: $\tau(x^\omega) = s \cdot y^{\omega-1} \cdot t$.*

These constitute our main ways to effectively evaluate the image of profinite words through transducers. Their use being quite ubiquitous in our study, we will rarely refer to these lemmata nominally.

3.3 The Syncing Lemma: Preservation Lemma applied to transducers

We apply the Preservation Lemma on transducers and deduce a slightly more combinatorial characterization of transducers describing continuous functions. This does not provide an immediate decidable criterion, but our decidability results will often rely on it. The goal of the forthcoming lemma is to decouple, when evaluating $s \cdot u \cdot t$ (with the notations of the Preservation Lemma), the behavior of the u part and that of the s, t part. This latter part will be tested against an *equalizer* set:

[3] In all the varieties we are interested in, one can easily modify any partial function into a total function while preserving its continuity properties.

▶ **Definition 8** (Equalizer set). Let $u, v \in \widehat{A^*}$. The *equalizer set* of u and v in $\mathcal{V}$ is:

$$\mathsf{Equ}\_\mathcal{V}(u,v) = \{(s,s',t,t') \in (A^*)^4 \mid s \cdot u \cdot t = \_\mathcal{V} s' \cdot v \cdot t'\} \ .$$

▶ Remark. The complexity of equalizer sets can be surprisingly high. For instance, letting $\mathcal{V}$ be the class of languages defined by $\{x^2 = x^3 \mid x \in A^*\}$, there is a profinite word u for which $\mathsf{Equ}\_\mathcal{V}(u,u)$ is undecidable. On the other hand, equalizer sets quickly become less complex for common varieties; for instance, Lemma 12 will provide a simple form for the equalizer sets of aperiodic supercancellative varieties.

▶ **Definition 9** (Input synchronization). Let $R, S \subseteq A^* \times B^*$. The *input synchronization* of R and S is defined as the relation over $B^* \times B^*$ obtained by synchronizing the first component of R and S: $R \bowtie S = \{(u,v) \mid (\exists s)[(s,u) \in R \wedge (s,v) \in S]\}$ $\big(= S \circ R^{-1}\big)$.

Naturally, the input synchronization of two rational functions is a rational relation.

▶ **Lemma 10** (Syncing Lemma). *Let τ be an unambiguous transducer from A^* to B^* and E a set of equations that defines $\mathcal{V}(A^*)$. The function τ is $\mathcal{V}$_continuous iff:*

1. *$\tau^{-1}(B^*) \in \mathcal{V}(A^*)$, and*
2. *For any $(u = v) \in E$, any states p, q, any $p' \in p.u$, and any $q' \in q.v$, and letting $u' = \tau_{p,p'}(u)$ and $v' = \tau_{q,q'}(v)$: $(\tau_{\bullet,p} \bowtie \tau_{\bullet,q}) \times (\tau_{p',\bullet} \bowtie \tau_{q',\bullet}) \subseteq \mathsf{Equ}\_\mathcal{V}(u',v')$.*

3.4 A profinite toolbox for the aperiodic setting

In this section, we provide a few lemmata pertaining to our study of aperiodic continuity. We show that the equalizer sets of aperiodic supercancellative varieties are well-behaved. Intuitively, the larger the varieties are, the more their nonempty equalizer sets will be similar to the identity. For instance, if $s \cdot x^\omega = \_\mathcal{A} x^\omega$, for words s and x, it should hold that s and x have the same primitive root. We first note the following easy fact that will only be used in this section; it is reminiscent of the notion of *equidivisibility*, studied in the profinite context by Almeida and Costa [2].

▶ **Lemma 11.** *Let u, v be profinite words over an alphabet A and $\mathcal{V}$ be a supercancellative variety. Suppose that there are $s, t \in A^*$ such that $u \cdot t = \_\mathcal{V} s \cdot v$, then there is a $w \in \widehat{A^*}$ such that $u = \_\mathcal{V} s \cdot w$ and $v = \_\mathcal{V} w \cdot t$. If moreover $u = v$ and $\mathcal{V}$ is aperiodic, then $u = \_\mathcal{V} s \cdot u \cdot t$.*

▶ **Lemma 12.** *Let u, v be profinite words over an alphabet A and $\mathcal{V}$ be an aperiodic supercancellative variety. Suppose $\mathsf{Equ}\_\mathcal{V}(u,v)$ is nonempty. There are words $x, y \in A^*$ and two pairs $\rho\_1, \rho\_2 \in (A^*)^2$ such that: $\mathsf{Equ}\_\mathcal{V}(u,v) = \Big(Id \cdot \big((x^*, x^*)\rho\_1^{-1}\big)\Big) \times \Big(\big(\rho\_2^{-1}(y^*, y^*)\big) \cdot Id\Big)$.*

▶ **Lemma 13.** *Let x, y be words. For every aperiodic supercancellative variety $\mathcal{V}$, it holds that $\mathsf{Equ}\_\mathcal{V}(x^\omega, y^\omega) = \mathsf{Equ}\_\mathcal{A}(x^\omega, y^\omega)$.*

▶ Remark. For two aperiodic supercancellative varieties $\mathcal{V}$ and $\mathcal{W}$, we could further show that if both $\mathsf{Equ}\_\mathcal{V}(u,v)$ and $\mathsf{Equ}\_\mathcal{W}(u,v)$ are nonempty, then they are equal, for any profinite words u, v. It may however happen that one equalizer set is empty while the other is not; for instance, with $u = (ab)^\omega$ and $v = (ab)^\omega \cdot a \cdot (ab)^\omega$, the equalizer set of u and v in $\mathcal{DA}$ is nonempty, while it is empty in $\mathcal{A}$.

4 Intermezzos

We present a few facts of independent interest on continuous rational functions. Through this, we develop a few examples, showing in particular how the Preservation and Syncing

Lemmata can be used to show (non)continuity. In a first part, we study when the structure of the transducer is relevant to continuity, and in a second, when the (non)inclusion of variety relates to (non)inclusion of the class of continuous rational functions.

4.1 Transducer structure and continuity

As noted by Reutenauer and Schützenberger [15, p. 231], there exist numerous natural varieties $\mathcal{V}$ for which any $\mathcal{V}$_realizable rational function is $\mathcal{V}$_continuous. Indeed:

▶ **Proposition 14.** *Let $\mathcal{V}$ be a variety of languages closed under inverse $\mathcal{V}$_realizable rational function. Any $\mathcal{V}$_realizable rational function is $\mathcal{V}$_continuous. This holds in particular for the varieties $\mathcal{A}, \mathcal{G}_{\text{sol}}$, and $\mathcal{G}$.*

▶ **Proposition 15.** *For $\mathcal{V} \in \{\mathcal{J}, \mathcal{L}, \mathcal{R}, \mathcal{DA}, \mathcal{A}_B, \mathcal{G}_{\text{nil}}, \mathcal{C}_{OM}\}$, there are $\mathcal{V}$_realizable rational functions that are not $\mathcal{V}$_continuous.*

The converse concern, that is, whether all $\mathcal{V}$_continuous rational functions are $\mathcal{V}$_realizable, was mentioned by Reutenauer and Schützenberger [15] for $\mathcal{V} = \mathcal{A}$.

▶ **Proposition 16.** *For $\mathcal{V} \in \{\mathcal{J}, \mathcal{L}, \mathcal{R}, \mathcal{DA}, \mathcal{A}, \mathcal{A}_B, \mathcal{C}_{OM}\}$, there are $\mathcal{V}$_continuous rational functions that are not $\mathcal{V}$_realizable.*

Proof. *(The aperiodic cases)* Let $A = \{a\}$, a unary alphabet. Consider the transducer τ that removes every second a: its minimal transducer not being a $\mathcal{A}$_transducer, it is not $\mathcal{A}$_realizable (this is a property of subsequential transducers [15]). However, all the unary languages of $\mathcal{V}$ are either finite or co-finite, and hence for any $L \in \mathcal{V}(A^*)$, $\tau^{-1}(L)$ is either finite or co-finite, hence belongs to $\mathcal{V}(A^*)$.

(The $\mathcal{A}_B$ and $\mathcal{C}_{OM}$ cases) Over $A = \{a, b\}$, define τ to map words w in aA^* to $(ab)^{|w|}$, and words w in bA^* to $(ba)^{|w|}$. Clearly, a and b cannot act commutatively on the transducer. Now $\tau(ab) = \_\mathcal{C}_{OM}\tau(ba)$, and moreover $\tau(x^\omega) = \_\mathcal{A}_B(ab)^\omega = \_\mathcal{A}_B1 = \tau(1)$, hence τ is continuous for both $\mathcal{A}_B$ and $\mathcal{C}_{OM}$ by the Preservation Lemma. ◀

We delay the positive answers to that question, namely for $\mathcal{G}_{\text{nil}}, \mathcal{G}_{\text{sol}}, \mathcal{G}$, to Corollary 27 as they constitute our main lever towards the decidability of continuity for these classes.

4.2 Variety inclusion and inclusion of classes of continuous functions

In this section, we study the consequence of variety (non)inclusion on the inclusion of the related classes of continuous rational functions. This is reminiscent of the notion of *heredity* studied by [12], where a function is $\mathcal{V}$_hereditarily continuous if it is $\mathcal{W}$_continuous for each subvariety $\mathcal{W}$ of $\mathcal{V}$. Variety noninclusion provides the simplest study case here:

▶ **Proposition 17.** *Let $\mathcal{V}$ and $\mathcal{W}$ be two varieties. If $\mathcal{V} \not\subseteq \mathcal{W}$ then there are $\mathcal{V}$_continuous rational functions that are not $\mathcal{W}$_continuous.*

The remainder of this section focuses on a dual statement:

If $\mathcal{V} \subsetneq \mathcal{W}$, are all $\mathcal{V}$_continuous rational functions $\mathcal{W}$_continuous?

We first focus on group varieties. Naturally, if 1. $\mathcal{V}$_continuous rational functions are $\mathcal{V}$_realizable and 2. $\mathcal{W}$_realizable rational functions are $\mathcal{W}$_continuous, this holds. Appealing to the forthcoming Corollary 27 for point 1 and Proposition 14 for point 2, we then get:

▶ **Proposition 18.** *For $\mathcal{V}, \mathcal{W} \in \{\mathcal{G}_{\text{nil}}, \mathcal{G}_{\text{sol}}, \mathcal{G}\}$ with $\mathcal{V} \subsetneq \mathcal{W}$, all $\mathcal{V}$_continuous rational functions are $\mathcal{W}$_continuous. This however fails for $\mathcal{V} = \mathcal{A}_B$ and for any $\mathcal{W} \in \{\mathcal{G}_{\text{nil}}, \mathcal{G}_{\text{sol}}, \mathcal{G}\}$.*

Proof. It remains to show the case $\mathcal{V} = \mathcal{A}_B$. This is in fact the same example as in the proof of Proposition 16, to wit, over $A = \{a, b\}$, the rational function τ that maps $w \in aA^*$ to $(ab)^{|w|}$, and words $w \in bA^*$ to $(ba)^{|w|}$. Indeed, we saw that this function is continuous for $\mathcal{A}_B$, but it holds that $\tau(a) = ab$ on the one hand, and $\tau(b^\omega a) = (ba)^\omega ba = \_\mathcal{W}ba$, but $ab \neq \_\mathcal{W}ba$. The Preservation Lemma then shows that τ is not continuous for $\mathcal{W}$. ◀

▶ **Proposition 19.** *All $\mathcal{A}_B$_continuous rational functions are $\mathcal{C}_{OM}$_continuous.*

We now turn to aperiodic varieties. For lesser expressive varieties, the property fails:

▶ **Proposition 20.** *For $\mathcal{V} \in \{\mathcal{J}, \mathcal{L}, \mathcal{R}\}$ and $\mathcal{W} \in \{\mathcal{L}, \mathcal{R}, \mathcal{DA}, \mathcal{A}\}$ with $\mathcal{V} \subsetneq \mathcal{W}$, there are $\mathcal{V}$_continuous rational functions that are not $\mathcal{W}$_continuous.*

▶ **Proposition 21.** *Any $\mathcal{DA}$_continuous rational function is $\mathcal{A}$_continuous.*

Proof. First note that both $\mathcal{DA}$ and $\mathcal{A}$ satisfy the hypotheses of Lemma 12. Consider a $\mathcal{DA}$_continuous rational function $\tau\colon A^* \to B^*$. By the Syncing Lemma, to show that it is $\mathcal{A}$_continuous, it is enough to show that 1. $\tau^{-1}(B^*) \in \mathcal{A}(A^*)$, and 2. That some input synchronizations of τ, based on equations of the form $x^\omega = \_\mathcal{A}x^{\omega+1}$, belong to an equalizer set of the form (by Lemma 7):

$$\mathsf{Equ}\_\mathcal{A}(\alpha \cdot y^\omega \cdot \beta,\ \alpha' \cdot z^\omega \cdot \beta') = \{(s, s', t, t') \mid (s \cdot \alpha,\ s' \cdot \alpha',\ \beta \cdot t,\ \beta' \cdot t') \in \mathsf{Equ}\_\mathcal{A}(y^\omega,\ z^\omega)\}\ .$$

Applying the Syncing Lemma on τ for the variety $\mathcal{DA}$, we get that point 1 is true, since $\tau^{-1}(B^*) \in \mathcal{DA}(A^*)$. Similarly, point 2 is true since $x^\omega = x^{\omega+1}$ is an equation of $\mathcal{DA}$, and Lemma 13 implies that the equalizer set of the equation above is the same in $\mathcal{DA}$ and $\mathcal{A}$. ◀

▶ **Proposition 22.** *There are nonrational functions that are continuous for both $\mathcal{DA}$ and Reg but are not $\mathcal{A}$_continuous.*

5 Deciding continuity for transducers

5.1 Deciding continuity for group varieties

Reutenauer and Schützenberger showed in [15] that a rational function is $\mathcal{G}$_continuous iff it is $\mathcal{G}$_realizable. Since this is proven effectively, it leads to the decidability of $\mathcal{G}$_continuity. In Proposition 14, we saw that the right-to-left statement also holds for $\mathcal{G}_{\text{sol}}$; we now show that the left-to-right statement holds for all group varieties $\mathcal{V}$ that contain $\mathcal{G}_{\text{nil}}$. As in [15], but with sensibly different techniques, we show that $\mathcal{V}$_continuous transducers are plurisubsequential. The Syncing Lemma will then imply that such transducers are $\mathcal{V}$_transducers. Both properties rely on the following normal form:

▶ **Lemma 23.** *Let τ be a transducer. An equivalent transducer τ' can be constructed by adjoining some codeterministic automaton to τ so that for any states p, q of τ':*

$$\Big[(\exists x, y)\big[\emptyset \neq (\tau'_{p,\bullet} \bowtie \tau'_{q,\bullet}) \subseteq (x, y) \cdot Id\big]\Big] \Rightarrow p = q\ .$$

Alternatively, the "dual" property can be ensured, adjoining a deterministic automaton to τ, so that for any states p, q of τ':

$$\Big[(\exists x, y)\big[\emptyset \neq (\tau'_{\bullet,p} \bowtie \tau'_{\bullet,q}) \subseteq Id \cdot (x, y)\big]\Big] \Rightarrow p = q\ .$$

▶ **Lemma 24.** *Let $\mathcal{V}$ be a variety of group languages that contains $\mathcal{G}_{\mathrm{nil}}$. For any $\mathcal{V}_$continuous unambiguous transducer τ, the transducer obtained by applying the dual of Lemma 23, then applying its first part, is a plurisubsequential $\mathcal{V}_$transducer.*

Proof. Write τ' for the result of the dual part of Lemma 23 on τ, and τ'' for the result of the first part of Lemma 23 on τ'. For these transducers, call a triple a states (p, q, q') a *fork* on a if from p, the transducer can go to q and q' reading one a, and there is a path from q to p reading only a's. Dually, a triple (q, q', p) is a *reverse fork* on a if the transducer can go from q and q' to p reading one a, and there is a path from p to q that reads only a. In both cases, the fork is *proper* if $q \neq q'$. We rely on two facts:

▶ **Fact 25.** *There are no proper forks or reverse forks in τ''.*

▶ **Fact 26.** *For any state p of τ'' and any letter a, it holds that $p \in p.a^\omega$.*

Consider a state p in τ'' and a letter a. As $p \in p.a^\omega$ by Fact 26, there is a cycle of a's on p. Call q the first state of that cycle. Next, let q' be such that (p, a, q') is a transition of τ''. Clearly, (p, q, q') forms a fork, hence by Fact 25, $q = q'$. Thus τ'' is plurisubsequential.

It remains to show that τ'' is a $\mathcal{V}_$transducer. To do so, consider an equation $u = \_\mathcal{V}v$, a state q of τ'', and let $p = q.u$ and $p' = q.v$. We show that $p = p'$, concluding the proof. We rely on the Syncing Lemma, since τ'' is $\mathcal{V}_$continuous; it ensures in particular that:

$$(\tau''_{\bullet,q} \bowtie \tau''_{\bullet,q}) \times (\tau''_{p,\bullet} \bowtie \tau''_{p',\bullet}) \subseteq \mathsf{Equ}\_\mathcal{V}(u', v') \quad \text{with } u' = \tau''_{q,p}(u), v' = \tau''_{q,p'}(v) \ . \tag{1}$$

Let (s, s, t_1, t_2) be in the left-hand side. It holds that $s \cdot u' \cdot t\_1 = \_\mathcal{V}s \cdot v' \cdot t\_2$, thus $u' \cdot t\_1 = \_\mathcal{V}v' \cdot t\_2$ (here and in the following, we derive equivalent equations by appealing to the fact that the *free group* is embedded, in a precise sense, in $\mathcal{V}$ [16, § 6.1.9]). Now consider another tuple (s', s', t_1', t_2') again in the left-hand side of Equation (1). It also holds that $u' \cdot t\_1' = \_\mathcal{V}v' \cdot t\_2'$, hence we obtain that $t\_1 \cdot t\_2^{-1} = \_\mathcal{V}t\_1' \cdot t\_2'^{-1}$. This is in turn equal in $\mathcal{V}$ to some $\alpha \cdot \beta^{-1}$ such that α and β are words that do not share the same last letter. This shows that $t\_1 = \alpha \cdot t$ and $t\_2 = \beta \cdot t$ for some word t, and similarly for t_1' and t_2'. More generally: $(\tau''_{p,\bullet} \bowtie \tau''_{p',\bullet}) \subseteq (\alpha, \beta) \cdot Id$, and the normal form of Lemma 23 thus shows that $p = p'$. ◀

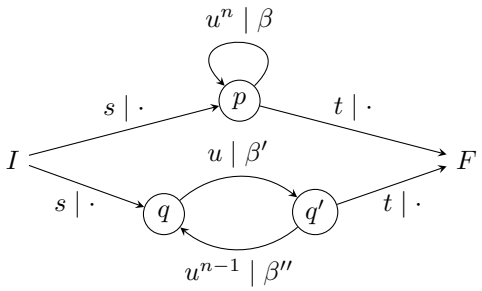
▶ **Corollary 27.** *For $\mathcal{V} \in \{\mathcal{G}_{\mathrm{nil}}, \mathcal{G}_{\mathrm{sol}}, \mathcal{G}\}$, any $\mathcal{V}_$continuous rational function is $\mathcal{V}_$realizable.*

▶ **Theorem 28.** *Let $\mathcal{V}$ be a variety of group languages that includes $\mathcal{G}_{\mathrm{nil}}$ and that is closed under inverse $\mathcal{V}_$realizable rational functions. It is decidable, given an unambiguous transducer, whether it realizes a $\mathcal{V}_$continuous function. This holds in particular for $\mathcal{G}_{\mathrm{sol}}$ and $\mathcal{G}$.*

5.2 Deciding continuity for aperiodic varieties

We saw in Section 4.1 that the approach of the previous section cannot work: there is no correspondence between continuity and realizability for aperiodic varieties. Herein, we use the Syncing Lemma to decide continuity in two main steps. First, note that all of our aperiodic varieties are defined by an infinite number of equations for each alphabet. The Syncing Lemma would thus have us check an infinite number of conditions; our first step is to reduce this to a finite number, which we stress through the forthcoming notion of "pertaining triplet" of states. Second, we have to show that the inclusion of the second point of the Syncing Lemma can effectively be checked. This will be done by simplifying this condition, and showing a decidability property on rational relations.

▶ **Definition 29.** A triplet of states (p, q, q') is *pertaining* if there are words s, u, t and an integer n such that:



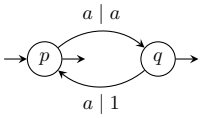
where $\cdot$ means "any word." Further, a pertaining triplet is *empty* if, in the above picture, $\beta = \beta'\beta'' = 1$ and *full* if both words are nonempty; it is *degenerate* if only one of β or $\beta'\beta''$ is empty.

It is called "pertaining" as the second point of the Syncing Lemma elaborates on properties of such a triplet, in particular, since $u^\omega = u^{\omega+1}$ is an equation of $\mathcal{A}$. The following characterization of $\mathcal{A}$_continuity is then made *without appeal* to equations or profinite words:

▶ **Lemma 30.** *A transducer $\tau\colon A^* \to B^*$ is $\mathcal{A}$_continuous iff all of the following hold:*

1. $\tau^{-1}(B^*) \in \mathcal{A}(A^*)$;
2. *For all full pertaining triplets (p, q, q'), there exist $x, y \in B^*$ and $\rho\_1, \rho\_2 \in (B^*)^2$ such that $\tau_{\bullet,p} \bowtie \tau_{\bullet,q} \subseteq Id \cdot \big((x^*, x^*)\rho\_1^{-1}\big)$ and $\tau_{p,\bullet} \bowtie \tau_{q',\bullet} \subseteq \big(\rho\_2^{-1}(y^*, y^*)\big) \cdot Id$;*
3. *For all empty pertaining triplets (p, q, q') it holds that $(\tau_{\bullet,p} \bowtie \tau_{\bullet,q}) \cdot (\tau_{p,\bullet} \bowtie \tau_{q',\bullet}) \subseteq Id$;*
4. *No pertaining triplet is degenerate.*

▶ **Example 31.** We show that the transducer of Proposition 16 is $\mathcal{A}$_continuous. Let τ be:



First, the function is total, hence the first point of Lemma 30 is verified. Second, there are no empty nor degenerate pertaining triplets, hence the third and fourth points are verified. Now the full pertaining triplets are $(p, p, p), (p, p, q), (q, q, q)$, and (q, q, p). We check that the pertaining triplet (p, p, q) verifies the second condition of Lemma 30, the other cases being similar or clear. The first half of the condition is immediate. Now $\tau_{p,\bullet} \bowtie \tau_{q,\bullet} = \{(a^{\lfloor n+1/2 \rfloor}, a^{\lfloor n/2 \rfloor}) \mid n \geq 0\}$ which verifies the condition.

We now show that the property of Lemma 30 is indeed decidable:

▶ **Proposition 32.** *It is decidable, given a rational relation $R \subseteq A^* \times A^*$, whether there is a word $x \in A^*$ and a pair $\rho \in (A^*)^2$, such that $R \subseteq Id \cdot \big((x^*, x^*)\rho^{-1}\big)$.*

▶ Remark. In general, the problem of deciding, given a rational relation R and a *recognizable* relation K, whether $R \subseteq Id \cdot K$, is undecidable. Indeed, testing $R \cap Id = \emptyset$ is undecidable [3], and equivalent to testing:

$$R \subseteq Id \cdot \big((A^+ \times \{1\}) \cup (\{1\} \times A^+) \cup \bigcup\nolimits\_{a \neq b \in A} (a \cdot A^* \times b \cdot A^*)\big) \ ,$$

the right-hand side being of the form $Id \cdot K$.

▶ **Theorem 33.** *It is decidable, given an unambiguous transducer, whether it realizes an $\mathcal{A}$_continuous function.*

The same approach, with carefully tweaked conditions, yields:

▶ **Theorem 34.** *For $\mathcal{V} = \mathcal{J}, \mathcal{R}, \mathcal{L}, \mathcal{DA}$, it is decidable, given an unambiguous transducer, whether it realizes a $\mathcal{V}$_continuous function.*

5.3 Deciding $\mathcal{C}om$- and $\mathcal{A}b$-continuity

The case of $\mathcal{C}om$ and $\mathcal{A}b$ is comparatively much simpler, in particular because these varieties are defined using a finite number of equations for each alphabet. However, the argument relies on different ideas:

▶ **Theorem 35.** *For $\mathcal{V} = \mathcal{C}om, \mathcal{A}b$, it is decidable, given an unambiguous transducer, whether it realizes a $\mathcal{V}$_continuous function.*

Proof. We apply the Syncing Lemma. Its first point is clearly decidable. We reduce its second point to decidable properties about semilinear sets (see, e.g., [7]). We also rely on the notion of Parikh image, that is, the mapping $\mathsf{Pkh}\colon A^* \to \mathbb{N}^A$ such that $\mathsf{Pkh}(w)$ maps $a \in A$ to the number of a's in the word w.

Since every $\mathcal{A}b$_continuous function is $\mathcal{C}om$_continuous (Proposition 19), the conditions to test for $\mathcal{A}b$_continuity are included in those for $\mathcal{C}om$_continuity—this can also be seen as a consequence of the fact that if u, v are words, $\mathsf{Equ}\_\mathcal{A}b(u, v) = \mathsf{Equ}\_\mathcal{C}om(u, v)$.

Let $\tau\colon A^* \to B^*$ be a given transducer. Consider an equation $ab = ba$ and four states p, p', q, q' of τ. Write $u = \tau_{p,p'}(ab)$ and $v = \tau_{q,q'}(ba)$. We ought to check, by the Syncing Lemma, the inclusion in $\mathsf{Equ}\_\mathcal{C}om(u, v) = \{(s, s', t, t') \mid s \cdot u \cdot t = \_\mathcal{C}om s' \cdot v \cdot t'\}$ of some input synchronization. Now this set is the set of (s, s', t, t') such that $\mathsf{Pkh}(s \cdot u \cdot t) = \mathsf{Pkh}(s' \cdot v \cdot t')$, and is thus defined by a simple semilinear property. The input synchronizations themselves, e.g., $\tau_{\bullet,p} \bowtie \tau_{\bullet,q}$, are rational relations, and their component-wise Parikh image is thus a semilinear set. Since the inclusion of semilinear sets is decidable, the inclusion of the second point of the Syncing Lemma is also decidable.

For $\mathcal{A}b$, we should additionally check the equations $a^\omega = 1$. The reasoning is similar. Consider three states (p, p', q), and write $x \cdot u^{\omega-1} \cdot y$ for $\tau_{p,p'}(a^\omega)$. By commutativity and the fact that $u^{\omega-1}$ acts as an inverse of u in the equations holding in $\mathcal{A}b$, we have that $(s, s', t, t') \in \mathsf{Equ}\_\mathcal{A}b(x \cdot u^{\omega-1} \cdot y, 1)$ iff $s \cdot t = \_\mathcal{A}b s' \cdot u \cdot t'$. This again reduces the inclusion of the second point of the Syncing Lemma to a decidable semilinear property. ◀

6 Discussion

We presented a study of continuity in functional transducers, on the one hand focused on general statements (Section 3), on the other hand on continuity for classical varieties. The heart of this contribution resides in decidability properties (Section 5), although we also addressed natural and related questions in a systematic way (Section 4). We single out two main research directions.

First, there is a sharp contrast between the genericity of the Preservation and Syncing Lemma and the technicality of the actual proofs of decidability of continuity. To which extent can these be unified and generalized? We know of two immediate extensions: 1. the generic results of Section 3 readily apply to Boolean algebras of languages closed under quotient, a relaxation of the conditions imposed on varieties, and 2. the varieties $\mathcal{G}\_p$ of languages recognized by p-groups can also be shown to verify Proposition 14 and Lemma 24, hence

$\mathcal{G}_p$_continuity is decidable for transducers. Beyond these two points, we do not know how to show decidability for $\mathcal{G}_{\text{nil}}$ (which is the *join* of the $\mathcal{G}_p$), and the surprising complexity of the equalizer sets for some Burnside varieties (e.g., the one defined by $x^2 = x^3$, see the Remark on page 7) leads us to conjecture that continuity may be undecidable in that case, hence that no unified way to show the decidability of continuity exists.

Second, the notion of continuity may be extended to more general settings. For instance, departing from regular languages, it can be noted that every recursive function is continuous for the class of recursive languages. Another natural generalization consists in studying $(\mathcal{V}, \mathcal{W})$_continuity, that is, the property for a function to map $\mathcal{W}$_languages to $\mathcal{V}$_languages by inverse image. This would provide more flexibility for a sufficient condition for cascades of languages (or stackings of circuits, or nestings of formulas) to be in a given variety.

Acknowledgment. We are deeply indebted to Jorge Almeida (in particular for the Remark on page 7), Luc Dartois, Bruno Guillon (in particular for the Remark on page 11), Ismaël Jecker, and Jean-Éric Pin for their insightful comments and kind help.

References

1 Jorge Almeida. Some algorithmic problems for pseudovarieties. *Publ. Math. Debrecen*, 54(1):531–552, 1999.

2 Jorge Almeida and Alfredo Costa. Equidivisible pseudovarieties of semigroups. *Publicationes Mathematicae Debrecen*, 90(3-4), 2017.

3 Jean Berstel. *Transductions and Context-Free Languages*, volume 38 of *Leitfäden der angewandten Mathematik und Mechanik LAMM*. Teubner, 1979.

4 Michaël Cadilhac, Andreas Krebs, Michael Ludwig, and Charles Paperman. A circuit complexity approach to transductions. In *MFCS*, pages 141–153, 2015. doi:10.1007/978-3-662-48057-1_11.

5 Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. First-order definability of rational transductions: An algebraic approach. In *LICS*, pages 387–396. ACM, 2016. doi:10.1145/2933575.2934520.

6 Mai Gehrke, Serge Grigorieff, and Jean-Eric Pin. Duality and equational theory of regular languages. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II – Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2008. doi:10.1007/978-3-540-70583-3_21.

7 Seymour Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, Inc., New York, NY, USA, 1966.

8 Jean-Éric Pin. Mathematical foundations of automata theory, 2016.

9 Jean-Éric Pin and Jacques Sakarovitch. Some operations and transductions that preserve rationality. In *Theoretical Computer Science*, pages 277–288. Springer, 1982. doi:10.1007/BFb0036488.

10 Jean-Éric Pin and Jacques Sakarovitch. Une application de la representation matricielle des transductions. *Theoretical Computer Science*, 35:271–293, 1985. doi:10.1016/0304-3975(85)90019-2.

11 Jean-Éric Pin and Pedro V. Silva. A topological approach to transductions. *Theoretical Computer Science*, 340(2):443–456, 2005. doi:10.1016/j.tcs.2005.03.029.

12 Jean-Éric Pin and Pedro V. Silva. On profinite uniform structures defined by varieties of finite monoids. *Int. J. of Algebra and Computation*, 21(01n02):295–314, 2011. doi:10.1142/S0218196711006170.

13 Jean-Éric Pin and Pedro V. Silva. On uniformly continuous functions for some profinite topologies. *Theoretical Computer Science*, 658, Part A:246–262, 2017. Formal Languages and Automata: Models, Methods and Application In honour of the 70th birthday of Antonio Restivo. doi:10.1016/j.tcs.2016.06.013.

14 Jean-Éric Pin and Pascal Weil. Profinite semigroups, Mal'cev products and identities. *Journal of Algebra*, 182(3):604–626, 1996.

15 Christophe Reutenaeur and Marcel-Paul Schützenberger. Variétés et fonctions rationnelles. *Theoretical Computer Science*, 145(1–2):229–240, July 1995. doi:10.1016/0304-3975(94)00180-Q.

16 Derek J. S. Robinson. *A Course in the Theory of Groups*. Springer, 2 edition, 1995.

17 Klaus Schneider. *Verification of Reactive Systems: Formal Methods and Algorithms*. SpringerVerlag, 2004.

18 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, 1994. doi:10.1007/978-1-4612-0289-9.

19 Pascal Tesson and Denis Thérien. Logic meets algebra: the case of regular languages. *Logical Methods in Computer Science*, 3(1), 2007.

A Universal Ordinary Differential Equation*

Olivier Bournez†[1] and Amaury Pouly[2]

1 **Ecole Polytechnique, LIX, Palaiseau Cedex, France**
bournez@lix.polytechnique.fr
2 **MPI-SWS, Saarbrücken, Germany**
pamaury@mpi-sws.org

Abstract

An astonishing fact was established by Lee A. Rubel (1981): there exists a fixed non-trivial fourth-order polynomial differential algebraic equation (DAE) such that for any positive continuous function φ on the reals, and for any positive continuous function $\epsilon(t)$, it has a $\mathcal{C}^\infty$ solution with $|y(t) - \varphi(t)| < \epsilon(t)$ for all t. Lee A. Rubel provided an explicit example of such a polynomial DAE. Other examples of universal DAE have later been proposed by other authors.

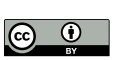
However, while these results may seem very surprising, their proofs are quite simple and are frustrating for a computability theorist, or for people interested in modeling systems in experimental sciences. First, the involved notions of universality is far from usual notions of universality in computability theory because the proofs heavily rely on the fact that constructed DAE does not have unique solutions for a given initial data. Indeed, in general a DAE may not have a unique solution, given some initials conditions. But Rubel's DAE *never* has a unique solution, even with a countable number of conditions of the form $y^{(k_i)}(a_i) = b_i$. This is very different from usual notions of universality where one would expect that there is clear unambiguous notion of evolution for a given initial data, for example as in computability theory. Second, the proofs usually rely on solutions that are piecewise defined. Hence they cannot be analytic, while analycity is often a key expected property in experimental sciences. Third, the proofs of these results can be interpreted more as the fact that (fourth-order) polynomial algebraic differential equations is a too loose a model compared to classical ordinary differential equations. In particular, one may challenge whether the result is really a universality result.

The question whether one can require the solution that approximates φ to be the unique solution for a given initial data is a well known open problem [Rubel 1981, page 2], [Boshernitzan 1986, Conjecture 6.2]. In this article, we solve it and show that Rubel's statement holds for polynomial ordinary differential equations (ODEs), and since polynomial ODEs have a unique solution given an initial data, this positively answers Rubel's open problem. More precisely, we show that there exists a **fixed** polynomial ODE such that for any φ and $\epsilon(t)$ there exists some initial condition that yields a solution that is ϵ-close to φ at all times.

The proof uses ordinary differential equation programming. We believe it sheds some light on computability theory for continuous-time models of computations. It also demonstrates that ordinary differential equations are indeed universal in the sense of Rubel and hence suffer from the same problem as DAEs for modelization: a single equation is capable of modelling any phenomenon with arbitrary precision, meaning that trying to fit a model based on polynomial DAEs or ODEs is too general (if it has a sufficient dimension).

1998 ACM Subject Classification G.1.7 Ordinary Differential Equations F.1.1 Models of Computation. F.1.3 Complexity Measures and Classes

* Full version at https://arxiv.org/abs/1702.08328.
† Olivier Bournez was partially supported by *ANR PROJECT RACAF.*



44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 116; pp. 116:1–116:14

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

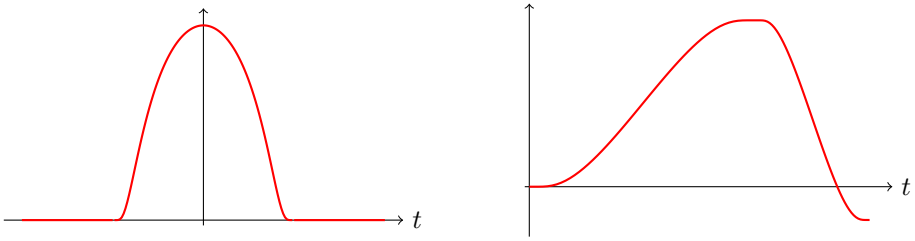


Figure 1 On left, graphical representation of function g. On right, two S-modules glued together.

Keywords and phrases Ordinary Differential Equations, Universal Differential Equations, Analog Models of Computation, Continuous-Time Models of Computation, Computability, Computational Analysis, Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.116

1 Introduction

A very astonishing result was established by Lee A. Rubel in 1981 [19]. There exists a universal fourth-order algebraic differential equation in the following sense.

▶ **Theorem 1** ([19]). *There exists a non-trivial fourth-order implicit differential algebraic equation*

$$P(y', y'', y''', y'''') = 0 \tag{1}$$

where P is a polynomial in four variables with integer coefficients, such that for any continuous function φ on $(-\infty, \infty)$ and for any positive continuous function $\epsilon(t)$ on $(-\infty, \infty)$, there exists a $\mathcal{C}^\infty$ solution y such that

$$|y(t) - \varphi(t)| < \epsilon(t)$$

for all $t \in (-\infty, \infty)$.

Even more surprising is the fact that Rubel provided an explicit example of such a polynomial P that is particularly simple:

$$\begin{aligned} 3y'^4y''y''''^2 \quad & -4y'^4y'''^2y'''' + 6y'^3y''^2y'''y'''' + 24y'^2y''^4y'''' \\ & -12y'^3y''y'''^3 - 29y'^2y''^3y'''^2 + 12y''^7 \qquad = 0 \end{aligned} \tag{2}$$

While this result looks very surprising at first sight, Rubel's proofs turns out to use basic arguments, and can be explained as follows. It uses the following classical trick to build $\mathcal{C}^\infty$ piecewise functions: let $g(t) = e^{-1/(1-t^2)}$ for $-1 < t < 1$, and $g(t) = 0$ otherwise. It is not hard to see that function g is C^∞ and Figure 1 shows that g looks like a "bump". Since it satisfies $\frac{g'(t)}{g(t)} = -\frac{2t}{(1-t^2)^2}$, then $g'(t)(1-t^2)^2 + g(t)2t = 0$ and $f(t) = \int_0^t g(u)du$ satisfies the polynomial differential algebraic equation $f''(1-t^2)^2 + f'(t)2t = 0$. Since this equation is homogeneous, it also holds for $af + b$ for any a and b. The idea is then to obtain a fourth order DAE that is satisfied by every function $y(t) = \gamma f(\alpha t + \beta) + \delta$, for all $\alpha, \beta, \gamma, \delta$. After some computations, Rubel obtained the universal differential equation (2).

Functions of the type $y(t) = \gamma f(\alpha t + \beta) + \delta$ generate what Rubel calls S-modules: a function that values A at a, B at b, is constant on $[a, a + \delta]$, monotone on $[a + \delta, b - \delta]$,

constant on $[b-\delta, b]$, by an appropriate choice of $\alpha, \beta, \gamma, \delta$. Summing S-modules corresponds to gluing then together, as is depicted in Figure 1. Note that finite, as well as infinite sums[1] of S-modules still satisfy the equation (2) and thus any piecewise affine function (and hence any continuous function) can be approximated by an appropriate sum of S-modules. This concludes Rubel's proof of universality.

As one can see, the proof turns out to be frustrating because the equation essentially allows any behavior. This may be interpreted as merely stating that differential algebraic equations is simply too lose a model. Clearly, a key point is that this differential equation does not have a unique solution for any given initial condition: this is the core principle used to glue a finite or infinite number of S-modules and to approximate any continuous function. Rubel was aware of this issue and left open the following question in [19, page 2].

> "It is open whether we can require in our theorem that the solution that approximates φ to be the unique solution for its initial data."

Similarly, the following is conjectured in [4, Conjecture 6.2].

> "Conjecture. There exists a non-trivial differential algebraic equation such that any real continuous function on $\mathbb{R}$ can be uniformly approximated on all of $\mathbb{R}$ by its real-analytic solutions"

The purpose of this paper is to provide a positive answer to both questions. We prove that a fixed polynomial ordinary differential equations (ODE) is universal in above Rubel's sense. At a high level, our proofs are based on ordinary differential equation programming. This programming is inspired by constructions from our previous paper [7]. Here, we mostly use this programming technology to achieve a very different goal and to provide positive answers to these above open problems.

We also believe they open some lights on computability theory for continuous-time models of computations. In particular, it follows that concepts similar to Kolmogorov complexity can probably be expressed naturally by measuring the complexity of the initial data of a (universal-) polynomial ordinary differential equations for a given function. We leave this direction for future work.

1.1 Related work and discussions

First, let us mention that Rubel's universal differential equation has been extended in several papers. In particular, Duffin proved in [12] that implicit universal differential equations with simpler expressions exists, such as $n^2 y''''y'^2 + 3n(1-n)y'''y''y' + (2n^2-3n+1)y''^3 = 0$ for any $n > 3$. The idea of [12] is basically to replace the $\mathcal{C}^\infty$ function g of [19] by some piecewise polynomial of fixed degree, that is to say by splines. Duffin also proves that considering trigonometric polynomials for function $g(x)$ leads to the universal differential equation $ny''''y'^2 + (2-3n)y'''y''y' + 2(n-1)y''^3 = 0$. This is done at the price of approximating function φ respectively by splines or trigonometric splines solutions which are $\mathcal{C}^n$ (and n can be taken arbitrary big) but not $\mathcal{C}^\infty$ as in [19]. Article [8] proposes another universal differential equation whose construction is based on Jacobian elliptic functions. Notice that [8] is also correcting some statements of [12].

[1] With some convergence or disjoint domain conditions.

All the results mentioned so far are concerned with approximations of continuous functions over the whole real line. Approximating functions over a compact domain seems to be a different (and somewhat easier for our concerns) problem, since basically by compactness, one just needs to approximate the function locally on a finite number of intervals. A 1986 reference survey discussing both approximation over the real line and over compacts is [4]. Recently, over compact domains, the existence of universal ordinary differential equation $\mathcal{C}^\infty$ of order 3 has been established in [11]: it is shown that for any $a < b$, there exists a third order $\mathcal{C}^\infty$ differential equation $y''' = F(y, y', y'')$ whose solutions are dense in $\mathcal{C}^0([a, b])$. Notice that this is not obtained by explicitly stating such an order 3 universal ordinary differential, and that this is a weaker notion of universality as solutions are only assumed to be arbitrary close over a compact domain and not all the real line. Order 3 is argued to be a lower bound for Lipschitzian universal ODEs [11].

Rubel's result has sometimes been considered to be related to be the equivalent, for analog computers, of the universal Turing machines. This includes Rubel's paper motivation given in [19, page 1]. We now discuss and challenge this statement.

Indeed, differential algebraic equations are known to be related to the General Purpose Analog Computer (GPAC) of Claude Shannon [20], proposed as a model of the Differential Analysers [9], a mechanical programmable machine, on which he worked as an operator. Notice that the original relations stated by Shannon in [20] between differential algebraic equations and GPACs have some flaws, that have been corrected later by [18] and [13]. Using the better defined model of GPAC of [13], it can be shown that functions generated by GPAC exactly correspond to polynomial ordinary differential equations. Some recent results have established that this model, and hence polynomial ordinary differential equations can be related to classical computability [5] and complexity theory [7].

However, we do not really follow the statement that Rubel's result is the equivalent, for analog computers, of the universal Turing machines. In particular, Rubel's notion of universality is completely different from the ones in computability theory. For a given initial data, a (deterministic) Turing machine has only one possible evolution. On the other hand, Rubel's equation does not dictate any evolution but rather some conditions that any evolution has to satisfy. In other words, Rubel's equation can be interpreted as the equivalent of an invariant of the dynamics of (Turing) machines, rather than a universal machine in the sense of classical computability.

Notice that while several results have established that (polynomial) ODEs are able to simulate the evolution of Turing machines (see e.g. [5, 15, 7]), the existence of a universal ordinary differential equation does not follow from them. To understand the difference, let us restate the main result of [15], of which [7] is a more advanced version for polynomial-time computable functions.

▶ **Theorem 2.** *A function $f : [a, b] \to \mathbb{R}$ is computable (in the framework of Computable Analysis) if and only if there exists some polynomials $p : \mathbb{R}^{n+1} \to \mathbb{R}^n$, $p_0 : \mathbb{R} \to \mathbb{R}$ with computable coefficients and $\alpha_1, \ldots, \alpha_{n-1}$ computable reals such that for all $x \in [a, b]$, the solution $y : [a, b] \to \mathbb{R}^n$ to the Cauchy problem*

$$y(0) = (\alpha_1, \ldots, \alpha_{n-1}, p_0(x)), \qquad y' = p(y)$$

satisfies that for all $t \geqslant 0$ that

$$|f(x) - y_1(t)| \leqslant y_2(t) \quad \text{and} \quad \lim_{t\to\infty} y_2(t) = 0.$$

Since there exists a universal Turing machine, there exists a "universal" polynomial ODE for computable functions. But there are major differences between Theorem 2 and the result

of this paper (Theorem 3). Even if we have a strong link between the Turing machines's configuration and the evolution of the differential equation, this is not enough to guarantee what the trajectory of the system will be at all times. Indeed, Theorem 2 only guarantees that $y_1(t) \to f(x)$ asymptotically. On the other hand, Theorem 3 guarantees the value of $y_1(t)$ at all times. Notice that our universality result also applies to functions that are not computable (in which case the initial condition is computable from the function but still not computable).

We would like to mention some implications for experimental sciences that are related to the classical use of ODEs in such contexts. Of course, we know that this part is less formal from a mathematical point of view, but we believe this discussion has some importance: A key property in experimental sciences, in particular physics is analyticity. Recall that a function is analytic if its is equal to its Taylor expansion in any point. It has sometimes been observed that "natural" functions coming from Nature are analytic, even if this cannot be a formal statement, but more an observation. We obtain a fixed universal polynomial ODEs, so in particular all its solution must be analytic[2], and it follows that universality holds even with analytic functions. All previous constructions mostly worked by gluing together $\mathcal{C}^\infty$ or $\mathcal{C}^n$ functions, and as it is well known "gluing" of analytic functions is impossible. We believe this is an important difference with previous works.

As we said, Rubel's proof can be seen as an indication that (fourth-order) polynomial implicit DAE is too loose model compared to classical ODEs, allowing in particular to glue solutions together to get new solutions. As observed in many articles citing Rubel's paper, this class appears so general that from an experimental point of view, it makes littles sense to try to fit a differential model because a single equation can model everything with arbitrary precision. Our result implies the same for polynomial ODEs since, for the same reason, a single equation of sufficient dimension can model everything.

Notice that our constructions have at the end some similarities with Voronin's theorem. This theorem states that Riemann's ζ function is such that for any analytic function $f(z)$ that is non-vanishing on a domain U homeomorphic to a closed disk, and any $\epsilon > 0$, one can find some real value t such that for all $z \in U$, $|\zeta(z + it) - f(z)| < \epsilon$. Notice that ζ function is a well-known function known not to be solution of any polynomial DAE (and consequently polynomial ODE), and hence there is no clear connexion to our constructions based on ODEs. We invite to read the post [17] in "Gödel's Lost Letter and P=NP" blog for discussions about potential implications of this surprising result to computability theory.

1.2 Formal statements

▶ **Theorem 3** (Universal PIVP). *There exists a* ***fixed*** *polynomial vector p in d variables such that for any functions $f \in C^0(\mathbb{R})$ and $\varepsilon \in C^0(\mathbb{R}, \mathbb{R}_{>0})$, there exists $\alpha \in \mathbb{R}^d$ such that there exists a unique solution $y : \mathbb{R} \to \mathbb{R}^d$ to $y(0) = \alpha$, $y' = p(y)$. Furthermore, this solution satisfies that $|y_1(t) - f(t)| \leqslant \varepsilon(t)$ for all $t \in \mathbb{R}$, and it is analytic.*

It is well-known that polynomial ODEs can be transformed into DAEs that have the same analytic solutions, see [10] for example. The following then follows for DAEs.

▶ **Theorem 4** (Universal DAE). *There exists a* ***fixed*** *polynomial p in $d+1$ variables such that for any functions $f \in C^0(\mathbb{R})$ and $\varepsilon \in C^0(\mathbb{R}, \mathbb{R}_{>0})$, there exists $\alpha_0, \ldots, \alpha_{d-1} \in \mathbb{R}$ such that*

[2] Which is not the case for polynomial DAEs.

there exists a unique ***analytic*** *solution* $y : \mathbb{R} \to \mathbb{R}$ *to* $y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(d-1)}(0) = \alpha_{d-1}$, $p(y, y', \ldots, y^d) = 0$. *Furthermore, this solution satisfies that* $|y(t) - f(t)| \leqslant \varepsilon(t)$ *for all* $t \in \mathbb{R}$.

▶ Remark. Notice that both theorems apply even when f is not computable. In this case, the initial condition(s) α exist but are not computable. We believe that α is always *computable from* f and ε, that is the mapping $(f, \varepsilon) \mapsto \alpha$ is computable in the framework of Computable Analysis, with an adequate representation of f, ε and α.

▶ Remark. Notice that we do not provide explicitly in this paper the considered polynomial ODE, nor its dimension d. But it can be derived by following the constructions. We currently estimate d to be more than three hundred following the precise constructions of this paper (but also to be very far from the optimal). We did not try to minimize d in the current paper, as we think our results are sufficiently hard to be followed in this paper for not beeing complicated by considerations about optimizations of dimensions.

▶ Remark. Both theorems are stated for *total functions* f and ε over $\mathbb{R}$. It trivially applies to any continuous partial function that can be extended to a continuous function over $\mathbb{R}$. In particular, it applies to any functions over $[a, b]$. It is not hard to see that it also applies to functions over (a, b) by rescaling $\mathbb{R}$ into (a, b) using the cotangent:

$$z(t) = y\left(-\cot\left(\tfrac{t-a}{b-a}\pi\right)\right) \quad \text{satisfies} \quad z'(t) = \phi'(t)p(z(t)), \quad \phi'(t) = \tfrac{\pi}{b-a}(1 + \phi(t)^2).$$

More complex domains such as $[a, b)$ and $(a, b]$ (with a possibly infinite) can also be obtain in a similar fashion.

2 Overview of the proof

A first a priori difficulty is that if one considers a fixed polynomial ODE $y' = p(y)$, one could think that the growth of its solutions is constrained by p and thus cannot be arbitrary. This would then prevent us from building a universal ODE simply because it could not grow fast enough. This fact is related to Emil Borel's conjecture in [3] (see also [16]) that a solution, *defined over* $\mathbb{R}$, to a system with n variables has growth bounded by roughly $e_n(x)$, the $n-$th iterate of exp. The conjecture is proved for $n = 1$ [3], but has been proven to be false for $n = 2$ in [21] and [2]. Bank [1] then adapted the previous counter-examples to provide a DAE whose non-unique increasing real-analytic solutions at infinity do not have any majorant. See the discussions (and Conjecture 6.1) in [4] for discussions about the growth of solutions of DAEs, and their relations to functions $e_n(x)$.

Thus, the first important part of this paper is to refine Bank's counter-example to build `fastgen`, a fast-growing function that satisfies even stronger properties. The second major ingredient is to be able to approximate a function with arbitrary precision everywhere. Since this is a difficult task, we use `fastgen` to our advantage to show that it is enough to approximate functions that are bounded and change slowly (think 1-Lipschitz, although the exact condition is more involved). That is to say, to deal with the case where there is no problem about the growth and rate of change of functions in some way. This is the purpose of the function `pwcgen` which can build arbitrary almost piecewise constant functions as long as they are bounded and change slowly.

It should be noted that the entire paper, we construct *generable functions* (in several variables) (see Section 3.1). For most of the constructions, we only use basic facts like the fact that generable functions are stable under arithmetic, composition and ODE solving. We know that generable functions satisfy polynomial partial equations and use this fact only at

the very end to show that the generable approximation that we have built, in fact, translates to a polynomial ordinary differential equation.

The rest of the paper is organized as follows. In Section 3, we recall some concepts and results from other articles. The main purpose of this section is to present Theorem 10. This theorem is the analog equivalent of doing an assignment in a periodic manner. Section 4 is devoted to `fastgen`, the fast-growing function. In Section 5, we show how to generate a sequence of dyadic rationals. In Section 6, we show how to generate a sequence of bits. In Section 7, we show how to leverage the two previous sections to generate arbitrary almost piecewise constant functions. Section 8 is then devoted to the proof of our main theorem.

3 Concepts and results from other articles

3.1 Generable functions

The following concept can be attributed to [20]: a function $f : \mathbb{R} \to \mathbb{R}$ is said to be a PIVP (Polynomial Initial Value Problem) function if there exists a system of the form $y' = p(y)$, where p is a (vector of) polynomial, with $f(t) = y_1(t)$ for all t, where y_1 denotes first component of the vector y defined in $\mathbb{R}^d$. We need in our proof to extend this concept to talk about multivariable functions. In [6], we introduced the following class, which can be seen as extensions of [14].

▶ **Definition 5** (Generable function)**.** Let $d, e \in \mathbb{N}$, I be an open and connected subset of $\mathbb{R}^d$ and $f : I \to \mathbb{R}^e$. We say that f is generable if and only if there exists an integer $n \geqslant e$, a $n \times d$ matrix p consisting of polynomials with coefficients in $\mathbb{R}$, $x_0 \in \mathbb{R}^d$, $y_0 \in \mathbb{R}^n$ and $y : I \to \mathbb{R}^n$ satisfying for all $x \in I$:

- $y(x_0) = y_0$ and $J_y(x) = p(y(x))$
- $f(x) = (y_1(x), \ldots, y_e(x))$

▶ y satisfies a polynomial differential equation[3],

▶ the components of f are components of y.

This class strictly generalizes functions generated by polynomial ODEs. Indeed, in the special case of $d = 1$ (the domain of the function has dimension 1), the above definition is equivalent to saying that $y' = p(y)$ for some polynomial p. The interested reader can read more about this in [6].

For the purpose of this paper, the reader only needs to know that the class of generable functions enjoys many stability properties that make it easy to create new functions from basic operations. Informally, one can add, subtract, multiply, divide and compose them at will, the only requirement is that the domain of definition must always be connected. In particular, the class of generable functions contains some common mathematical functions:

- (multivariate) polynomials,
- trigonometric functions: sin, cos, tan, etc,
- exponential and logarithm: exp, ln,
- hyperbolic trigonometric functions: sinh, cosh, tanh.

Two famous examples of functions that are *not* in this class are the ζ and Γ, we refer the reader to [6] and [14] for more information.

A nontrivial fact is that generable functions are always analytic. This property is well-known in the one-dimensional case but is less obvious in higher dimensions, see [6] for more details. Moreover, generable functions satisfy the following crucial properties.

[3] J_y denotes the Jacobian matrix of y.

▶ **Lemma 6** (Closure properties of generable functions). *Let $f :\subseteq \mathbb{R}^d \to \mathbb{R}^n$ and $g :\subseteq \mathbb{R}^e \to \mathbb{R}^m$ be generable functions. Then $f + g$, $f - g$, fg, $\frac{f}{g}$ and $f \circ g$ are generable*[4].

▶ **Lemma 7** (Generable functions are closed under ODE). *Let $d \in \mathbb{N}$, $J \subseteq \mathbb{R}$ an interval, $f :\subseteq \mathbb{R}^d \to \mathbb{R}^d$ generable, $t_0 \in J$ and $y_0 \in \operatorname{dom} f$. Assume there exists $y : J \to \operatorname{dom} f$ satisfying*

$$y(t_0) = y_0 \qquad y'(t) = f(y(t))$$

for all $t \in J$, then y is generable (and unique).

In fact, generable functions satisfy the stronger (albeit more obscure) theorem

▶ **Theorem 8** (Generable functions are closed under ODE). *Let $d, n \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$, $t_0 \in \mathbb{R}$, $(J_\alpha)_{\alpha\in\Omega}$ a family of open intervals containing t_0 and $G : \Omega \to \mathbb{R}^n$, $F :\subseteq \mathbb{R}^d \to \mathbb{R}^d$ generable. Assume that $X = \{(\alpha, t) : \alpha \in \Omega, t \in J_\alpha\}$ is an open connected set and that there exists $f : X \to \operatorname{dom} F$ satisfying*

$$f(\alpha, t_0) = G(\alpha) \qquad \frac{\partial f}{\partial t}(\alpha, t) = F(f(\alpha, t))$$

for all $\alpha \in \Omega$ and $t \in J_\alpha$. Then f is generable (and unique).

3.2 Helper functions and constructions

We mentioned earlier that a number of common mathematical functions are generable. However, for our purpose, we will need less common functions that one can consider to be programming gadgets. One such operation is rounding (computing the nearest integer). Note that, by construction, generable functions are analytic and in particular must be continuous. It is thus clear that we cannot build a perfect rounding function and in particular we have to compromise on two aspects:

- we cannot round numbers arbitrarily close to $n + \frac{1}{2}$ for $n \in \mathbb{Z}$: thus the function takes a parameter λ to control the size of the "zone" around $n + \frac{1}{2}$ where the function does not round properly,
- we cannot round without error: thus the function takes a parameters μ that controls how good the approximation must be.

▶ **Lemma 9** (Round, [6]). *There exists a generable function* round *such that for any $n \in \mathbb{Z}$, $x \in \mathbb{R}$, $\lambda > 2$ and $\mu \geqslant 0$:*

- *if $x \in \left[n - \frac{1}{2}, n + \frac{1}{2}\right]$ then $|\operatorname{round}(x, \mu, \lambda) - n| \leqslant \frac{1}{2}$,*
- *if $x \in \left[n - \frac{1}{2} + \frac{1}{\lambda}, n + \frac{1}{2} - \frac{1}{\lambda}\right]$ then $|\operatorname{round}(x, \mu, \lambda) - n| \leqslant e^{-\mu}$.*

The other very useful operation is the analog equivalent of a discrete assignment, done in a periodic manner. More precisely, we consider a particular class of ODEs

$$y'(t) = \operatorname{pereach}(t, \phi(t), y(t), g(t))$$

adapted from the constructions of [7].

This equation alternates between two behaviors, for all $n \in \mathbb{N}$.

- During $J_n = [n, n + \frac{1}{2}]$, it performs $y(t) \to \overline{g}$ where $\min_{t\in J_n} g(t) \leqslant \overline{g} \leqslant \max_{t\in J_n} g(t)$. So in particular, if $g(t)$ is almost constant over this time interval, then it is essentially $y(t) \to g$. Then ϕ controls how good the convergence is: the error is of the order of $e^{-\phi}$.

[4] With the obvious dimensional condition associated with each operation.

- During $J'_n = [n+\frac{1}{2}, n+1]$, the systems tries to keep y constant, ie $y' \approx 0$. More precisely, the system enforces that $|y'(t)| \leqslant e^{-\phi(t)}$.

▶ **Theorem 10** (Periodic reach). *There exists a generable function* $\mathrm{pereach} : \mathbb{R}^2_{\geqslant 0} \times \mathbb{R}^2 \to \mathbb{R}$ *such that for any* $I = [n, n+1]$ *with* $n \in \mathbb{N}$, $y_0 \in \mathbb{R}$, $\phi, \psi \in C^0(I, \mathbb{R}_{\geqslant 0})$ *and* $g \in C^0(I, \mathbb{R})$, *the unique solution to*

$$y(n) = y_0, \qquad y'(t) = \psi(t)\,\mathrm{pereach}(t, \phi(t), y(t), g(t))$$

exists over I.

- *If there exists* $\bar{g} \in \mathbb{R}$ *and* $\eta \in \mathbb{R}_{\geqslant 0}$ *such that* $|g(t) - \bar{g}| \leqslant \eta$ *for all* $t \in [n, n+\frac{1}{2}]$, *then* $|y(t) - \bar{g}| \leqslant \eta + \exp\left(-\int_n^t \psi(u)\phi(u)du\right)$ *whenever* $\int_n^t \psi(u)\phi(u)du \geqslant 1$ *for all* $t \in [n, n+\frac{1}{2}]$, *and* $|y(t) - \bar{g}| \leqslant \max(\eta, |y(n) - \bar{g}|)$ *for all* $t \in [n, n+\frac{1}{2}]$ *without condition.*
- *For all* $t \in [n+\frac{1}{2}, n]$, $|y(t) - y(n+\frac{1}{2})| \leqslant \int_{n+\frac{1}{2}}^t \psi(u)\exp(-\phi(u))\,du$.

In particular, the first item implies that $y(t) \geqslant \min_{u\in[n,t]} g(t) - \exp\left(-\int_n^t \phi(u)du\right)$ *whenever* $\int_n^t \phi(u)du \geqslant 1$ *for all* $t \in [n, n+\frac{1}{2}]$, *and* $y(t) \geqslant \min\left(y(n), \min_{u\in[n,t]} g(t)\right)$.

4 Generating fast growing functions

Our construction crucially relies on our ability to build functions of arbitrary growth. At the end of this section, we obtain a function `fastgen` with a straightforward specification: for any infinite sequence $a_0, a_1, \ldots$ of positive numbers, we can find a suitable $\alpha \in \mathbb{R}$ such that $\mathrm{fastgen}(\alpha, n) \geqslant a_n$ for all $n \in \mathbb{N}$. Furthermore, we can ensure that $\mathrm{fastgen}(\alpha, \cdot)$ is increasing. Notice, and this is the key point, that the definition of `fastgen` is independent of the sequence a: a single generable function (and thus differential system) can have arbitrary growth by simply tweaking its initial value.

Our construction builds on the following lemma proved by [1], based on an example of [2]. The proof essentially relies on the function $\frac{1}{2-\cos(x)-\cos(\alpha x)}$ which is generable and well-defined for all positive x if α is irrational. By carefully chosing α, we can make $\cos(x)$ and $\cos(\alpha x)$ simultaneously arbitrary close to 1.

▶ **Lemma 11** ([1]). *There exists a positive nondecreasing generable function* g *and an absolute constant* $c > 0$ *such that for any increasing sequence* $a \in \mathbb{N}^{\mathbb{N}}$ *with* $a_n \geqslant 2$ *for all* n, *there exists* $\alpha \in \mathbb{R}$ *such that* $g(\alpha, \cdot)$ *is defined over* $[1, \infty)$ *and for any* $n \in \mathbb{N}$ *and* $t \geqslant 2\pi b_n$, $g(\alpha, t) \geqslant c a_n$ *where* $b_n = \prod_{k=0}^{n-1} a_k$.

Essentially, this lemma proves that there exists a function g such that for any $n \in \mathbb{N}$, $g(\alpha, a_0 a_1 \cdots a_{n-1}) \geqslant a_n$. Note that this is not quite what we are aiming for: the function g is indeed $\geqslant a_n$ but at times $a_0 a_1 \cdots a_{n-1}$ instead of n. Since $a_0 a_1 \cdots a_{n-1}$ is a very big number, we need to "accelerate" g so that it reaches this values faster. This is a chicken-and-egg problem because to accelerate g, we need to build a fast growing function. We now try to explain how to solve this problem. Consider the following sequence:

$$x_0 = a_0, \qquad x_{n+1} = x_n g(x_n).$$

Then observe that

$$x_1 = x_0 g(x_0) = a_0 g(a_0) \geqslant a_0 a_1, \qquad x_2 = x_1 g(x_1) \geqslant a_0 a_1 g(a_0 a_1) \geqslant a_0 a_1 a_2, \quad \ldots$$

It is not hard to see that $x_n \geqslant a_0 a_1 \cdots a_n \geqslant a_n$. We then use our generable gadget of Section 3.2 to simulate this discrete sequence with a differential equation. Intuitively, we

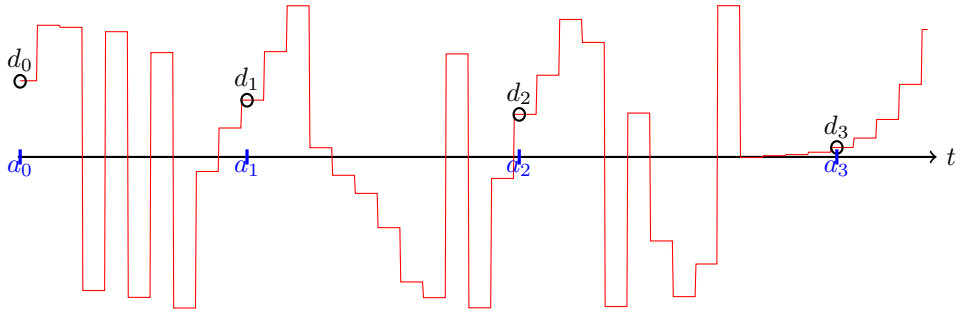


Figure 2 Graph of dygen for $d_0 = 2^{-1}$, $d_1 = 2^{-3} + 2^{-1}$, $d_2 = 2^{-5} + 2^{-2}$ and $d_3 = 2^{-4}$ (other values ignored) assuming that $\delta = 9$. We get that $a_0 = 0$, $a_1 = 10$, $a_2 = 22$, $a_3 = 36$.

build a differential equation such that the solution y satisfies $y(n) \approx x_n$. More precisely, we use two variables y and z such that over $[n, n + 1/2]$, $z' \approx 0$ and $y(t) \to zg(z)$ and over $[n+1/2, n+1]$, $y' \approx 0$ and $z(t) \to y$. Then if $y(n) \approx z(n) \approx x_n$ then $y(n+1) \approx z(n+1) \approx x_{n+1}$.

▶ **Theorem 12.** *There exists* $\Gamma \subseteq \mathbb{R}$ *and a positive generable function* $\mathrm{fastgen} : \Gamma \times \mathbb{R}_{\geqslant 0} \to \mathbb{R}$ *such that for any* $x \in \mathbb{R}^{\mathbb{N}}_{\geqslant 0}$*, there exists* $\alpha \in \Gamma$ *such that for any* $n \in \mathbb{N}$ *and* $t \in \mathbb{R}_{\geqslant 0}$*,*

$$\mathrm{fastgen}(\alpha, t) \geqslant x_n \qquad \text{if } t \geqslant n.$$

Furthermore, $\mathrm{fastgen}(\alpha, \cdot)$ *is nondecreasing.*

5 Generating a sequence of dyadic rationals

A major part of the proof requires to build a function to approximate arbitrary numbers over intervals $[n, n + 1]$. Ideally we would like to build a function that gives x_0 over $[0, 1]$, x_1 over $[1, 2]$, etc. Before we get there, we solve a somewhat simpler problem by making a few assumptions:

- we only try to approximate dyadic numbers, *i.e.* numbers of the form $m2^{-p}$, and furthermore we only approximate with error 2^{-p-3},
- if a dyadic number has size p, meaning that it can be written as $m2^{-p}$ but not $m'2^{-p+1}$ then it will take a time interval of p units to approximate: $[k, k + p]$ instead of $[k, k + 1]$,
- the function will only approximate the dyadics over intervals $[k, k + \frac{1}{2}]$ and not $[k, k + 1]$.

This processus is illustrated in Figure 2: given a sequence $d_0, d_1, \ldots$ of dyadics, there is a corresponding sequence $a_0, a_1, \ldots$ of times such that the function approximate d_k over $[a_k, a_k + \frac{1}{2}]$ within error 2^{-p_k} where p_k is the size of d_k. The theorem contains an explicit formula for a_k that depends on some absolute constant δ.

Let $\mathbb{D}_p = \{m2^{-p} : m \in \{0, 1, \ldots, 2^p - 1\}\}$ and $\mathbb{D} = \bigcup_{n\in\mathbb{N}} \mathbb{D}_p$ denote the set of dyadic rationals in $[0, 1)$. For any $q \in \mathbb{D}$, we define its *size* by $\mathfrak{L}(q) = \min\{p \in \mathbb{N} : q \in \mathbb{D}_p\}$.

▶ **Theorem 13.** *There exists* $\delta \in \mathbb{N}_{>0}$*,* $\Gamma \subseteq \mathbb{R}^2$ *and a generable function* $\mathrm{dygen} : \Gamma \times \mathbb{R}_{\geqslant 0} \to \mathbb{R}$ *such that for any dyadic sequence* $q \in \mathbb{D}^{\mathbb{N}}$*, there exists* $(\alpha, \beta) \in \Gamma$ *such that for any* $n \in \mathbb{N}$*,*

$$|\mathrm{dygen}(\alpha, \beta, t) - q_n| \leqslant 2^{-\mathfrak{L}(q_n)-3} \qquad \text{for any } t \in [a_n, a_n + \tfrac{1}{2}]$$

where $a_n = \sum_{k=0}^{n-1} (\mathfrak{L}(q_k) + \delta)$*. Furthermore,* $|\mathrm{dygen}(\alpha, \beta, t)| \leqslant 1$ *for all* α, β *and* t*.*

6 Generating a sequences of bits

We saw in the previous section how to generate a dyadic generator. Unfortunately, we saw that it generates dyadic d_n at times a_n, whereas we would like to get d_n at time n for our approximation. Our approach is to build a signal generator that will be high exactly at times a_n. Each the signal will be high, the system will copy the value of the dyadic generator to a variable and wait until the next signal. Since the signal is binary, we only need to generate a sequence of bits. Note that this theorem has a different flavour from the dyadic generator: it generates a more restrictive set of values (bits) but does so much better because we have better control of the timing and we can approximate the bits with arbitrary precision.

▶ Remark. Although it is possible to define `bitgen` using `dygen`, it does not, in fact, gives a shorter proof but definitely gives a more complicated function.

▶ **Theorem 14.** *There exists* $\Gamma \subseteq \mathbb{R}$ *and a generable function* $\operatorname{bitgen} : \Gamma \times \mathbb{R}^2_{\geqslant 0} \to \mathbb{R}$ *such that for any bit sequence* $b \in \{0,1\}^{\mathbb{N}}$*, there exists* $\alpha_b \in \Gamma$ *such that for any* $\mu \in \mathbb{R}_{\geqslant 0}$*,* $n \in \mathbb{N}$ *and* $t \in [n, n + \frac{1}{2}]$*,*

$$|\operatorname{bitgen}(\alpha_b, \mu, t) - b_n| \leqslant e^{-\mu}.$$

Furthermore, $|\operatorname{bitgen}(\alpha, \mu, t)| \leqslant 1$ *for all* α, μ *and* t*.*

7 Generating an almost piecewise constant function

We have already explained the main intuition of this section in previous sections. Using the dyadic generator and the bit generator as a signal, we can construct a system that "samples" the dyadic at the right time and then holds this value still until the next dyadic. In essence, we just described an almost piecewise constant function. This function still has a limitation: its rate of change is small so it can only approximate slowly changing functions.

▶ **Theorem 15.** *There exists an absolute constant* $\delta \in \mathbb{N}$*,* $p \in \mathbb{N}$*,* $\Gamma \subseteq \mathbb{R}^p$ *and a generable function* $\operatorname{pwcgen} : \Gamma \times \mathbb{R}_{\geqslant 0} \to \mathbb{R}$ *such that for any dyadic sequence* $q \in \mathbb{D}^{\mathbb{N}}$ *then there exists* $\alpha \in \Gamma$ *such that for any* $n \in \mathbb{N}$*,*

$$|\operatorname{pwcgen}(\alpha, t) - q_n| \leqslant 2^{-\mathfrak{L}(q_n)} \qquad \text{for any } t \in [a_n + \tfrac{1}{2}, a_{n+1}]$$

and[5]

$$\operatorname{pwcgen}(\alpha, t) \in \left[\operatorname{pwcgen}(\alpha, a_n), \operatorname{pwcgen}(\alpha, a_n + \tfrac{1}{2})\right] \qquad \text{for any } t \in [a_n, a_n + \tfrac{1}{2}]$$

where $a_n = \sum_{k=0}^{n-1} (\delta + \mathfrak{L}(q_k))$*.*

8 Proof of the main theorem

The proof works in several steps. First we show that using an almost constant function, we can approximate functions that are bounded and change very slowly. We then relax all these constraints until we get to the general case. In the following, we only consider total functions over $\mathbb{R}$. See Remark on page 6 for more details.

[5] With the convention that $[a, b] = [\min(a, b), \max(a, b)]$.

▶ **Definition 16** (Universality)**.** Let $I \subseteq \mathbb{R}$ and $\mathcal{C} \subseteq C^0(I) \times C^0(I, \mathbb{R}_{>0})$. We say that the universality property holds for $\mathcal{C}$ if there exists $d \in \mathbb{N}$ and a generable function $\mathfrak{u}$ such that for any $(f, \varepsilon) \in \mathcal{C}$, there exists $\alpha \in \mathbb{R}^d$ such that

$$|\mathfrak{u}(\alpha, t) - f(t)| \leqslant \varepsilon(t) \qquad \text{for any } t \in \mathrm{dom}(f).$$

▶ **Lemma 17.** *There exists a constant $c > 0$ such that the universality property holds for all (f, ε) on $\mathbb{R}_{\geqslant 0}$ such that for all $t \in \mathbb{R}_{\geqslant 0}$:*

- *ε is decreasing and $-\log_2 \varepsilon(t) \leqslant c' + t$ for some constant c',*
- *$f(t) \in [0, 1]$,*
- *$|f(t) - f(t')| \leqslant c\varepsilon(t+1)$ for all $t' \in [t, t+1]$.*

Proof Sketch. This is essentially a application of `pwcgen` with a small twist. Indeed the bound on f guarantees that dyadic rationals are enough. The bound on the rate of change of f guarantees that a single dyadic can provide an approximation for a long enough time. And the bound on ε guarantees that we do not need too many digits for the approximations. ◀

▶ **Lemma 18.** *The universality property holds for all (f, ε) on $\mathbb{R}_{\geqslant 0}$ such that f and ε are differentiable, ε is decreasing and $f(t) \in [0, 1]$ for all $t \in \mathbb{R}_{\geqslant 0}$.*

Proof Sketch. Consider $F = f \circ h^{-1}$ and $E = \varepsilon \circ h^{-1}$ where h is a fast-growing function like `fastgen`. Then the faster h grows, the slower E and F change and thus we can apply Lemma 17 to (F, E). We recover an approximation of f from the approximation of F. ◀

▶ **Lemma 19.** *The universality property holds for all (f, ε) on $\mathbb{R}_{\geqslant 0}$ such that f is differentiable and ε is decreasing.*

Proof Sketch. Consider $F = \frac{1}{2} + \frac{f}{h}$ and $\varepsilon = \frac{f}{h}$ where h is a fast-growing function like `fastgen`. By taking h big enough, we can ensure that $F(t) \in [0, 1]$ and apply Lemma 18 to (F, E). We then recover an approximation of f from the approximation of F. ◀

▶ **Lemma 20.** *The universality property holds for all continuous (f, ε) on $\mathbb{R}_{\geqslant 0}$.*

Proof Sketch. Observe that the set of differential functions is dense in the set of continuous functions and apply Lemma 19. ◀

▶ **Lemma 21.** *The universality property holds for all continuous (f, ε) on $\mathbb{R}$.*

Proof Sketch. Let f^+ be the approximating of f over $\mathbb{R}_{\geqslant 0}$, using Lemma 20. Then we extend and modify f^+ to $\mathbb{R}$ in such a way that the approximation is still good over $\mathbb{R}_{\geqslant 0}$ but the function almost vanishes over $(-\infty, 0]$. We then do the same to $t \mapsto f(-t) - f^+(-t)$ and sum the two functions. ◀

We can now show the main theorem.

Proof of Theorem 3. Lemma 21 gives a generable function $\mathfrak{u}$. There exists α such that

$$|\mathfrak{u}(\alpha, t) - f(t)| \leqslant \varepsilon(t).$$

And since $\mathfrak{u}$ is generable, $t \mapsto \mathfrak{u}(\alpha, t)$ satisfies a PIVP. ◀

References

1 Steven B. Bank. Some results on analytic and meromorphic solutions of algebraic differential equations. *Advances in Mathematics*, 15(1):41 – 62, 1975. doi:10.1016/0001-8708(75)90124-3.

2 N. M. Basu, S. N. Bose, and T. Vijayaraghavan. A simple example for a theorem of vijayaraghavan. *Journal of the London Mathematical Society*, s1-12(4):250–252, 1937. doi:10.1112/jlms/s1-12.48.250.

3 Emile Borel. Mémoire sur les séries divergentes. *Annales Scientifiques de l'Ecole Normale Supérieure*, 16:9–136, 1899.

4 Michael Boshernitzan. Universal formulae and universal differential equations. *Annals of mathematics*, 124(2):273–291, 1986.

5 Olivier Bournez, Manuel L. Campagnolo, Daniel S. Graça, and Emmanuel Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, 23(3):317–335, June 2007. doi:10.1016/j.jco.2006.12.005.

6 Olivier Bournez, Daniel S. Graça, and Amaury Pouly. On the functions generated by the general purpose analog computer. *CoRR*, abs/1602.00546, 2016. URL: http://arxiv.org/abs/1602.00546.

7 Olivier Bournez, Daniel S. Graça, and Amaury Pouly. Polynomial Time corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length. The General Purpose Analog Computer and Computable Analysis are two efficiently equivalent models of computations. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 109:1–109:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.109.

8 Keith Briggs. Another universal differential equation. *arXiv preprint math/0211142*, 2002.

9 V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.

10 D. C. Carothers, G. E. Parker, J. S. Sochacki, and P. G. Warne. Some properties of solutions to polynomial systems of differential equations. *Electron. J. Diff. Eqns.*, 2005(40), April 2005.

11 Etienne Couturier and Nicolas Jacquet. Construction of a universal ordinary differential equation $\rfloor^{\infty}$ of order 3. *arXiv preprint arXiv:1610.09148*, 2016.

12 Richard J. Duffin. Rubel's universal differential equation. *Proceedings of the National Academy of Sciences*, 78(8):4661–4662, 1981.

13 Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664, 2003.

14 D. S. Graça, J. Buescu, and M. L. Campagnolo. Computational bounds on polynomial differential equations. *Appl. Math. Comput.*, 215(4):1375–1385, 2009.

15 D. S. Graça, M. L. Campagnolo, and J. Buescu. Computability with polynomial differential equations. *Adv. Appl. Math.*, 40(3):330–349, 2008.

16 G. H. Hardy. Some results concerning the behaviour at infinity of a real and continuous solution of an algebraic differential equation of the first order. *Proceedings of the London Mathematical Society*, 2(1):451–468, 1912.

17 R. J. Lipton and K. W. Regan. The amazing zeta code. Post on Blog "Gödel's Lost Letter and P=NP", https://rjlipton.wordpress.com/2012/12/04/the-amazing-zeta-code/, December 4, 2012.

18 M. B. Pour-El. Abstract computability and its relations to the general purpose analog computer. *Trans. Amer. Math. Soc.*, 199:1–28, 1974.

19 L. A. Rubel. A universal differential equation. *Bulletin of the American Mathematical Society*, 4(3):345–349, May 1981.

20 C. E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.

21 T. Vijayaraghavan. Sur la croissance des fonctions définies par les équations différentielles. *CR Acad. Sci. Paris*, 194:827–829, 1932.

Regular Separability of Parikh Automata

Lorenzo Clemente[*1], Wojciech Czerwiński[†2], Sławomir Lasota[‡3], and Charles Paperman[4]

1 **University of Warsaw, Warsaw, Poland**
`l.clemente@mimuw.edu.pl`
2 **University of Warsaw, Warsaw, Poland**
`wczerwin@mimuw.edu.pl`
3 **University of Warsaw, Poland**
`sl@mimuw.edu.pl`
4 **University of Tübingen, Tübingen, Germany**
`charles.paperman@gmail.com`

Abstract

We investigate a subclass of languages recognized by vector addition systems, namely languages of nondeterministic Parikh automata. While the regularity problem (is the language of a given automaton regular?) is undecidable for this model, we surprisingly show decidability of the regular separability problem: given two Parikh automata, is there a regular language that contains one of them and is disjoint from the other? We supplement this result by proving undecidability of the same problem already for languages of visibly one counter automata.

1998 ACM Subject Classification D.2.2 [Design Tools and Techniques] Petri Nets, F.1.1 [Theory of Computation] Models of Computation, F.2.2 [Nonnumerical Algorithms and Problems] Computations on Discrete Structures, F.3.1 [Specifying and Verifying and Reasoning about Programs] Mechanical verification

Keywords and phrases Regular separability problem, Parikh automata, integer vector addition systems, visible one counter automata, decidability, undecidability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.117

1 Introduction

We investigate separability problems for languages of finite words. We say that a language U is *separated from* a language V *by* S if $U \subseteq S$ and $V \cap S = \emptyset$. In the sequel we also often say that U *and* V are *separated by* S. For two families of languages $\mathcal{F}$ and $\mathcal{G}$, the $\mathcal{F}$ *separability problem for* $\mathcal{G}$ asks for two given languages $U, V \in \mathcal{G}$ whether U is separated from V by some language from $\mathcal{F}$. The same notion of separability makes clearly sense if $\mathcal{F}$ and $\mathcal{G}$ are classes of sets of vectors instead of classes of languages. In this paper, we consider the case where $\mathcal{F}$ are regular languages and $\mathcal{G}$ languages recognized by Parikh automata, and the case where $\mathcal{F}$ are the unary sets and $\mathcal{G}$ the semilinear sets.

Motivation. Separability is a classical problem in theoretical computer science. It was investigated most extensively in the area of formal languages, for $\mathcal{G}$ being the family of all regular word languages. Since regular languages are effectively closed under complement, the

[*] Partially supported by the Polish National Science Centre grant 2016/21/B/ST6/01505.
[†] Partially supported by the Polish National Science Centre grant 2016/21/D/ST6/01376.
[‡] Partially supported by the Polish National Science Centre grant 2016/21/B/ST6/01505.

$\mathcal{F}$ separability problem is a generalization of the $\mathcal{F}$ characterization problem, which asks whether a given language belongs to $\mathcal{F}$. Indeed, $L \in \mathcal{F}$ if and only if L is separated from its complement by some language from $\mathcal{F}$. Separability problems for regular languages attracted recently a lot of attention, which resulted in establishing the decidability of $\mathcal{F}$ separability for various families $\mathcal{F}$ such as the piecewise testable languages [7, 15] (recently generalized to finite ranked trees [9]), the locally and locally threshold testable languages [14], the languages definable in first order logic [17], and the languages of certain higher levels of the first order hierarchy [16], among others.

Separability of nonregular languages attracted little attention till now. The reasons for this may be twofold. First, for regular languages one can use standard algebraic tools, like syntactic monoids, and indeed most of the results have been obtained with the help of such techniques. Second, some strong intractability results have been known already since 70's, when Szymanski and Williams proved that regular separability of context-free languages is undecidable [18]. Later Hunt [10] generalized this result for every class $\mathcal{F}$ closed under finite boolean combinations and containing all languages of the form $w\Sigma^*$ for $w \in \Sigma^*$. This is a very weak condition, so it seemed that nothing nontrivial can be done outside regular languages with respect to separability problems. Furthermore, Szymanski and Williams's negative result has recently been strengthened by considering two incomparable subclasses of pushdown automata. First, Kopczyński has shown that regular separability is undecidable for languages of visibly pushdown automata [13], and then Czerwiński and Lasota have shown that the same problem is undecidable for one counter automata [6].

On the positive side, piecewise testable separability has been shown decidable for context-free languages, languages of vector addition systems (VAS languages), and some other classes of languages [8]. Another surprising result has been recently obtained by Czerwiński and Lasota [6] who show that regular separability is decidable (and PSPACE-complete) for languages recognized by one counter nets (i.e., one counter automata without zero test). Notice that in all these examples regularity (resp. piecewise testability) is undecidable, but regular (resp. piecewise testable) separability *is* decidable, and until recently there were not many results of this kind.

Finally, in [5] we have shown decidability of *unary separability* of reachability sets of vector addition systems (VASes). By *unary sets* we mean Parikh images of commutative regular languages, and thus the latter problem is equivalent to commutative regular separability of (commutative closures of) VAS languages. The decidability status of the regular separability problem for the whole class of VAS languages remains open.

Our contribution. This paper is a continuation of the line of research trying to understand the regular separability problem for language classes beyond regular languages. We report a further progress towards solving the open problem mentioned above by providing a positive decidability result and a new negative undecidability result: As our first (positive) result, we show decidability of the regular separability problem for the subclass of VAS languages where we allow negative counter values during a run. This class of languages is also known as languages of *integer VASSes*, and it admits many different characterizations; for instance, it coincides with languages of *one-way reversal-bounded counter machines* [11], *Parikh automata* [12] (cf. also [2, Proposition 11]), which in turn are equivalent to the very similar model of *constrained automata* [3]. In this paper, we present our results in terms of constrained automata, but given the similarity with Parikh automata (and in light of their equivalence), we overload the name Parikh automata for both models.

Notice that PA languages are not closed under complement, and thus our decidability result about regular separability does not imply decidability of the regularity problem (is

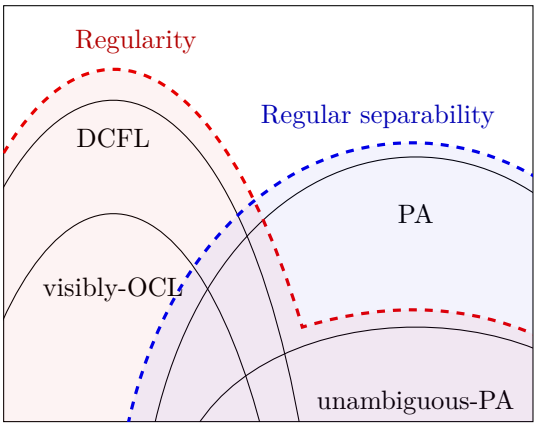


Figure 1 The regularity and the regular separability problems.

the language of a given Parikh automaton regular?). Moreover, the regularity problem for PA languages is actually *undecidable* [2][1], which makes our decidability result one of few instances where regularity is undecidable but regular separability is decidable; cf. Fig 1.

Parikh automata are finite nondeterministic automata where accepting runs are further restricted to satisfy a semilinear condition on the multiset of transitions appearing in the run. Our decidability result is actually stated in the more general setting of $\mathcal{C}$-*Parikh automata*, where $\mathcal{C} \subseteq \bigcup_{d\in\mathbb{N}} \mathcal{P}(\mathbb{N}^d)$ is a class of sets of vectors used as an acceptance condition. We prove that the regular separability problem for languages of $\mathcal{C}$-Parikh automata effectively reduces to the *unary* separability problem for the class $\mathcal{C}$ itself, provided that $\mathcal{C}$ is effectively closed under inverse images of affine functions. Two prototypical classes $\mathcal{C}$ satisfying the latter closure condition are semilinear sets and VAS reachability sets. Moreover, unary separability of semilinear sets is known to be decidable [4], and as recalled before the same result has recently been extended to VAS reachability sets [5]. As a consequence of our reduction, we deduce decidability of regular separability of $\mathcal{C}$-Parikh automata languages where the acceptance condition $\mathcal{C}$ can be instantiated to either the semilinear sets, or the VAS reachability sets.

We complement our decidability result by a new negative undecidability result subsuming simultaneously Kopczyński's undecidability for visibly pushdown languages [13] and Czerwiński and Lasota's undecidability of one counter languages [6]: We show that regular separability is undecidable for (deterministic[2]) visibly one counter languages. Inside the proof we use the result from [6], but actually in order to only reprove [13] it would be sufficient to use the old work by Szymanski and Williams [18].

2 Preliminaries

Vectors. A set $S \subseteq \mathbb{N}^d$ is *linear* if there exist a *base* $b \in \mathbb{N}^d$ and *periods* $p_1, \ldots, p_k \in \mathbb{N}^d$ s.t. $S = \{b + n_1p_1 + \ldots + n_kp_k \mid n_1, \ldots, n_k \in \mathbb{N}\}$, and it is *semilinear* if it is a finite union of linear sets. For a vector $v \in \mathbb{N}^d$ and $i \in \{1, \ldots, d\}$, let $v[i]$ denote its i-th coordinate. For $n \in \mathbb{N}$, we say that two vectors $x, y \in \mathbb{N}^d$ are n-*unary equivalent*, written $x \equiv_n y$, if for every coordinate $i \in \{1, \ldots, d\}$ it holds $x[i] \equiv y[i] \mod n$ and moreover $x[i] \leq n \iff y[i] \leq n$. A

[1] Later shown decidable for unambiguous PA [3].
[2] Determinism here is irrelevant because this class can be determinized.

set $S \subseteq \mathbb{N}^d$ is *unary* if for some n, S is a union of equivalence classes of $\equiv_n$. Intuitively, to decide membership in a unary set S it is enough to count on every coordinate exactly up to some threshold n, and modulo n for values larger than n. Unary sets are semilinear.

Let $\Sigma = \{a_1, \ldots, a_k\}$ be a totally ordered alphabet. For a word $w \in \Sigma^*$ and a letter $a_i \in \Sigma$, by $\#_{a_i}(w)$ we denote the number of letters a_i in w. The *Parikh image* of a word $w \in \Sigma^*$ is the vector $\Pi(w) = (\#_{a_1}(w), \ldots, \#_{a_k}(w)) \in \mathbb{N}^k$. The *Parikh image* of a language $L \subseteq \Sigma^*$ is $\Pi(L) = \{\Pi(w) \mid w \in L\}$, the set of Parikh images of all words belonging to L.

Parikh automata. A *nondeterministic finite automaton with ε-transitions* (ε-NFA) $\mathcal{A} = (Q, I, F, T)$ over a finite alphabet Σ consists of a finite set of states Q, two distinguished subsets of initial and final states $I, F \subseteq Q$, and a set of transitions $T \subseteq Q \times \Sigma_\varepsilon \times Q$, where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. A *nondeterministic Parikh automaton*[3] is a pair $(\mathcal{A}, S)$ consisting of an ε-NFA $\mathcal{A}$ and a semilinear set $S \subseteq \mathbb{N}^d$, where $d = |T|$ is the number of transitions of $\mathcal{A}$. Notice that we allow ε-transitions in the definition of Parikh automata. A *run* of a Parikh automaton over a word $w = a_1 \ldots a_n \in \Sigma^*$ is a sequence of transitions $\rho = t_1 \ldots t_n \in T^*$, where $t_i = (q_{i-1}, a_i, q_i)$, starting in an initial state q_0. A run ρ is *accepting* if its ending state q_n is final and $\Pi(\rho) \in S$ (we assume here that the set of transitions T is totally ordered). The language of a Parikh automaton, denoted $L(\mathcal{A}, S)$, contains all words w admitting an accepting run; it is thus a subset of the language $L(\mathcal{A})$ of the underlying ε-NFA.

One can generalize Parikh automata by using some other family of vector sets in the place of semilinear sets. For a class $\mathcal{C} \subseteq \bigcup_{d \in \mathbb{N}} \mathcal{P}(\mathbb{N}^d)$ of vector sets, a *$\mathcal{C}$-Parikh automaton* is a pair $(\mathcal{A}, S)$, where $\mathcal{A}$ is an ε-NFA and $S \in \mathcal{C}$. The language $L(\mathcal{A}, S)$ is defined as above.

A $\mathcal{C}$-Parikh automaton $(\mathcal{A}, S)$ is *deterministic* if the underlying automaton $\mathcal{A}$ is so; here, we assume that a deterministic automaton does not have ε-transitions. The languages of (non)deterministic $\mathcal{C}$-Parikh automata are shortly called (non)deterministic $\mathcal{C}$-Parikh languages below.

3 Main results

We call a function $f : \mathbb{N}^k \to \mathbb{N}^\ell$ *affine* if it is of the form $f(v) = Mv + u$ for an integer non-negative matrix M of dimension $\ell \times k$ and a vector $u \in \mathbb{N}^\ell$. In a special case when $u = 0$ we call the function f *linear*. A class of vector sets $\mathcal{C} \subseteq \bigcup_{d \in \mathbb{N}} \mathcal{P}(\mathbb{N}^d)$ is called *robust* if it fulfills the following two conditions:

- $\mathcal{C}$ is effectively closed under inverse images of affine functions,
- the unary separability problem is decidable for $\mathcal{C}$.

Our first main result is decidability of the regular separability problem for $\mathcal{C}$-Parikh automata.

▶ **Theorem 1.** *The regular separability problem is decidable for $\mathcal{C}$-Parikh automata, for every robust class $\mathcal{C}$ of vector sets.*

The proof of Theorem 1 is split into two parts. In Section 4 we provide a reduction of the regular separability problem for *nondeterministic* $\mathcal{C}$-Parikh automata to the same problem for *deterministic* ones; this step is crucial for understanding how the regular separability problem differs from the regularity problem, which does not admit a similar reduction. Then in Section 5 we reduce the regular separability problem for deterministic $\mathcal{C}$-Parikh automata to the unary separability problem for vector sets in $\mathcal{C}$.

[3] This is the same as *constrained automata* from [3].

In Section 6 we consider two instantiations of the class $\mathcal{C}$. First, taking $\mathcal{C}$ to be the semilinear sets we derive decidability for (ordinary) Parikh automata. Second, we consider the class $\mathcal{C}_{\text{SEC-VAS}}$ of *sections of* reachability sets of VASes (detailed definitions are deferred to Section 6), which allows us to obtain decidability for $\mathcal{C}_{\text{SEC-VAS}}$-Parikh automata. Note that the latter model properly extends Parikh automata.

Before proceeding with the rest of the proof for our decidability result, we present a generic reduction of the regular separability problem from which we can immediately derive a new undecidability result, which is our second main contribution.

3.1 A generic reduction

We observe that regular separability of homomorphic images of a class of languages $\mathcal{G}$ reduces to regular separability for $\mathcal{G}$ itself (cf. Lemma 2 below). Since nondeterministic Parikh automata are the homomorphic image of deterministic ones, we reduce regular separability for the nondeterministic class to the deterministic one (shown in Sec. 4); together with the decidability result for the deterministic class presented in Sec. 5, this proves Theorem 1.

Our reduction can also be used to derive undecidability results. Since context-free languages are the homomorphic image of (deterministic) visibly pushdown languages (cf. [1, Theorem 5.2]), and since regular separability is undecidable for the former class [18], we concisely reprove the recent LICS'16 result by Kopczyński [13] about undecidability of regular separability of visibly pushdown languages. Moreover, this result can be further strengthened to (deterministic) visibly one counter languages using the same observation and the recent result by Czerwiński and Lasota [6] about undecidability of regular separability for one counter automata (cf. Sec. 3.2).

We now present our generic reduction. Given two alphabets Σ and Γ, a *homomorphism* is a function $h : \Sigma \to \Gamma^*$ which extends homomorphically to a function from Σ^* to Γ^*, and thus to languages. For $\mathcal{G}$ a class of languages and $\mathcal{H}$ a class of homomorphisms, let $\mathcal{H}(\mathcal{G})$ be the class of languages obtained by applying some homomorphism from $\mathcal{H}$ to some language in $\mathcal{G}$.

▶ **Lemma 2.** *If $\mathcal{G}$ and $\mathcal{H}(\mathcal{G})$ are effectively closed under inverse images of homomorphisms from $\mathcal{H}$, then the regular separability problem in $\mathcal{H}(\mathcal{G})$ reduces to the same problem in $\mathcal{G}$.*

In statements of this form, "effective" means that for given finite computational model representing $L \in \mathcal{G}$ and $h \in \mathcal{H}$, one can effectively find a representation with a finite computational model for $h^{-1}(L) \in \mathcal{G}$, and similarly for $\mathcal{H}(\mathcal{G})$. The reduction above is a consequence of the following fundamental relationship between separators and (inverse) images of functions. (We do not exploit the further structure of homomorphisms here.)

▶ **Lemma 3.** *Let $L \subseteq \Sigma^*, K \subseteq \Gamma^*$ be two languages, and let $h : \Sigma^* \to \Gamma^*$ be a function.*
1. *If R separates $h(L)$ and K, then $h^{-1}(R)$ separates L and $h^{-1}(K)$.*
2. *If R separates L and $h^{-1}(K)$, then $h(R)$ separates $h(L)$ and K.*

Proof. The proof is elementary and it is given for completeness. For the first point, $L \subseteq h^{-1}(R)$ follows from the inclusion $h(L) \subseteq R$ since $L \subseteq h^{-1}(h(L))$, and the disjointness of $h^{-1}(R)$ and $h^{-1}(K)$ follows from disjointness of R and K. For the second point, the inclusion $h(L) \subseteq h(R)$ follows by the inclusion $L \subseteq R$, and the disjointness of $h(R)$ and K follows from the disjointness of R and $h^{-1}(K)$. ◀

Since regular languages are closed under images and inverse images of homomorphisms, we immediately obtain the following corollary.

▶ **Corollary 4.** *Let h be a homomorphism. Languages $h(L), K$ are regular separable if, and only if, $L, h^{-1}(K)$ are so.*

Since regular languages are closed under complement, the regular separability problem is in fact symmetric. Combining this observation with the corollary above, we can now prove correctness of our generic reduction.

Proof of Lemma 2. Let $h(L), K$ be two languages in $\mathcal{H}(\mathcal{G})$. By Corollary 4, regular separability for $h(L), K$ is the same as for $L, h^{-1}(K)$. Since $\mathcal{H}(\mathcal{G})$ is closed under inverse images by assumption, $h^{-1}(K)$ equals the image $g(K_1)$ of language K_1 in $\mathcal{G}$ for some g from $\mathcal{H}$. We have thus reduced to regular separability for $L, g(K_1)$, where now both L and K_1 are in $\mathcal{G}$. Since regular languages are closed under complement, regular separability for $L, g(K_1)$ is the same for $g(K_1), L$. Applying once more Corollary 4, the latter statement is equivalent to regular separability for $K_1, g^{-1}(L)$. Since $\mathcal{G}$ closed under inverse images by assumption, $g^{-1}(L)$ is itself in $\mathcal{G}$. Since every step was effective, this concludes the proof. ◀

3.2 A new undecidability result

A *one counter automaton* is a finite-state device manipulating a single natural counter, which can be incremented, decremented, and tested for zero; it is *visible* if the input symbol uniquely determines which counter operation will be performed. Therefore, languages recognized by visible one counter automata are a strict subclass of visibly pushdown languages [1]. It was recently proved that regular separability for one counter automata is undecidable [6], which is incomparable with undecidability for visibly pushdown languages [13].

As a consequence of Lemma 2 we obtain undecidability of regular separability for visible one counter automata, which is our second main result, strengthening both [6] and [13].

▶ **Theorem 5.** *Regular separability of languages recognised by (deterministic) visible one counter automata is undecidable.*

Let $\mathcal{G}$ be the class of languages recognized by visible one counter automata, and let $\mathcal{H}$ be the class of letter-to-letter (non-erasing) homomorphisms, i.e., functions of the form $h : \Sigma \to \Gamma$. In order to apply Lemma 2, it suffices to show that languages recognized by one counter automata are the effective homomorphic image of those recognized by the visible subclass, and that both classes are effectively closed under inverse images of letter-to-letter homomorphisms. We begin with the second result.

▶ **Lemma 6.** *One counter languages and visibly one counter languages are effectively closed under inverse images of letter-to-letter homomorphisms.*

Proof. Given one counter automaton $\mathcal{A}$ over Σ and a letter-to-letter homomorphism $h : \Gamma \to \Sigma$, one computes an automaton $\mathcal{B}$ over Γ of the same kind s.t. $L(\mathcal{B}) = h^{-1}(L(\mathcal{A}))$ as follows. The automaton $\mathcal{B}$ is obtained by replacing a transition reading $a \in \Sigma$ in $\mathcal{A}$ by corresponding transitions reading $b \in \Gamma$ *performing the same counter operation*, for every $b \in h^{-1}(a)$. Is it easy to check that $L(\mathcal{B}) = h^{-1}(L(\mathcal{A}))$, as required. Moreover, if $\mathcal{A}$ was visible, since the counter operation is preserved, $\mathcal{B}$ will be visible too. ◀

Proof of Theorem 5. It remains to show that one counter languages are the effective homomorphic image of visible one counter languages. This is easy to show. Let $L \subseteq \Sigma^*$ be a one counter language. Each symbol $a \in \Sigma$ is split into three symbols $a_{\mathsf{inc}}, a_{\mathsf{dec}}, a_{=0?}$. The corresponding homomorphism h just forgets the new annotation, i.e., $h(a_{\mathsf{inc}}) = h(a_{\mathsf{dec}}) = h(a_{=0?}) = a$; notice that h is letter-to-letter and non-erasing. Counter operations for the

new automaton are made visible by replacing an increment operation over a by the same operation over a_{inc}, and similarly for decrements and tests. Clearly, we obtain a visible one counter automaton recognizing a language M s.t. $L = h(M)$. Thus, by Lemma 2, Lemma 6, and the undecidability of regular separability of one counter languages [6, Theorem 2], we obtain that regular separability for visibly one counter languages is undecidable. ◀

4 From nondeterministic to deterministic PA

The aim of this section is to prove the following lemma:

▶ **Lemma 7.** *If $\mathcal{C}$ is effectively closed under inverse images of linear mappings, then the regular separability problem of nondeterministic $\mathcal{C}$-Parikh automata effectively reduces to the same problem for deterministic ones.*

As a consequence of Lemma 7, we can focus on separability of deterministic PA languages in the rest of the paper. Let $\mathcal{G}$ be the class of deterministic $\mathcal{C}$-Parikh automata languages, and let $\mathcal{H}$ be the class of *letter-to-letter erasing homomorphism*, i.e., functions of the form $h : \Sigma \to (\Gamma \cup \{\varepsilon\})$ extended homomorphically to $\Sigma^* \to \Gamma^*$. The proof of the lemma follows immediately from Lemma 2 once we prove that nondeterministic languages are the effective images of deterministic ones (cf. Lemma 8), and that both classes are closed under inverse images (cf. Lemma 9). In the rest of the section, we assume that the class $\mathcal{C}$ is closed under inverse images of linear mappings, which is the case for a robust class $\mathcal{C}$ (cf. Sec. 3).

▶ **Lemma 8.** *Every nondeterministic $\mathcal{C}$-Parikh language is the effective image of a letter-to-letter erasing homomorphism of a deterministic $\mathcal{C}$-Parikh language.*

Proof. Fix a nondeterministic $\mathcal{C}$-Parikh automaton $(\mathcal{A}, S)$ over the alphabet Σ, and let T be the set of transitions of $\mathcal{A}$. Consider the letter-to-letter erasing homomorphism $h : T \to (\Sigma \cup \{\varepsilon\})$ that maps a transition (p, a, q) to a. Let $(\mathcal{B}, S)$ be the deterministic $\mathcal{C}$-Parikh automaton over the alphabet T which is obtained from $\mathcal{A}$ by relabelling every transition of $t = (p, a, q) \in T$ of $\mathcal{A}$ as a (unique) transition (p, t, q) of $\mathcal{B}$. Notice that the acceptance condition of $\mathcal{B}$ is the same as that for $\mathcal{A}$, since we only relabelled transitions. One easily verifies that $L(\mathcal{A}, S) = h(L(\mathcal{B}, S))$, as required. ◀

▶ **Lemma 9.** *Deterministic and nondeterministic $\mathcal{C}$-Parikh languages are effectively closed under inverse images of letter-to-letter erasing homomorphisms.*

Proof. Given a deterministic $\mathcal{C}$-Parikh automaton $(\mathcal{A}, S)$ over Σ and a letter-to-letter erasing homomorphism $h : \Gamma \to (\Sigma \cup \{\varepsilon\})$, one computes a deterministic $\mathcal{C}$-Parikh automaton $(\mathcal{B}, T)$ s.t. $L(\mathcal{B}, T) = h^{-1}(L(\mathcal{A}, S))$ as follows. The automaton $\mathcal{B}$ is obtained by replacing every transition (p, a, q) in $\mathcal{A}$ by transitions (p, b, q), one for every $b \in h^{-1}(a)$. Moreover, each state p in the automaton $\mathcal{B}$ has a self-loop (p, b, p) for every $b \in h^{-1}(\varepsilon)$. The constraint $T \in \mathcal{C}$ is the inverse image of S under the linear function obtained by counting a transition (p, b, q) as a transition $(p, h(b), q)$ if $h(b) \neq \varepsilon$, and by counting (p, b, q) as zero (i.e., ignoring it) otherwise. Finally, the constraint T, and hence also the automaton $(\mathcal{B}, T)$ can be computed. Is it easy to check that $L(\mathcal{B}, T) = h^{-1}(L(\mathcal{A}, S))$, as required. Moreover, if $\mathcal{A}$ is deterministic, and h is a function, then the resulting automaton $\mathcal{B}$ is also deterministic. ◀

5 Regular separability reduces to unary separability

In this section we reduce regular separability of deterministic $\mathcal{C}$-Parikh languages to unary separability of vector sets in $\mathcal{C}$.

▶ **Lemma 10.** *Let $\mathcal{C}$ be a class of vectors effectively closed under inverse images of affine mappings. The regular separability problem for deterministic $\mathcal{C}$-Parikh automata reduces to the unary separability problem for vector sets in $\mathcal{C}$.*

The rest of this section is devoted to the proof of the lemma. Let $L_1, L_2 \subseteq \Sigma^*$ be languages of deterministic $\mathcal{C}$-Parikh automata $(\mathcal{A}_1, S_1)$ and $(\mathcal{A}_2, S_2)$, respectively. There are three steps:

1. As the first step, we show that w.l.o.g. we may assume $\mathcal{A}_1 = \mathcal{A}_2$.
2. In the second step, we partition Σ^* into finitely many regular languages $K_1, \ldots, K_m$ and we reduce regular separability of L_1 and L_2 to regular separability of $L_1 \cap K_i$ and $L_2 \cap K_i$ for every $i \in \{1, \ldots, m\}$. These subproblems turn out to be easier than the general one, due to the additional structural information encoded in the languages K_i's.
3. In the last step, we reduce regular separability of $L_1 \cap K_i$ and $L_2 \cap K_i$ to unary separability of vector sets in $\mathcal{C}$.

Step 1: Unifying the underlying automaton. As the input languages are subsets of regular languages recognised by their underlying finite automata, $L_1 = L(\mathcal{A}_1, S_1) \subseteq L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2, S_2) \subseteq L(\mathcal{A}_2)$, it is enough to consider separability of L_1 and L_2 *inside* the intersection of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$:

▶ **Proposition 1.** *The languages L_1 and L_2 are regular separable if, and only if, the languages $L_1 \cap L(\mathcal{A}_2)$ and $L_2 \cap L(\mathcal{A}_1)$ are so.*

Proof. The "only if" direction is trivial as every language separating L_1 and L_2 separates $L_1 \cap L(\mathcal{A}_2)$ and $L_2 \cap L(\mathcal{A}_1)$ as well. For the opposite direction, we observe that if a regular language S separates $L_1 \cap L(\mathcal{A}_2)$ and $L_2 \cap L(\mathcal{A}_1)$, then $S' = (S \cap L(\mathcal{A}_1)) \cup \overline{L(\mathcal{A}_2)}$ is a regular language separating L_1 and L_2. ◀

Let $\mathcal{A}$ be the product automaton of $\mathcal{A}_1$ and $\mathcal{A}_2$, and thus $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. It is deterministic since both $\mathcal{A}_1$ and $\mathcal{A}_2$ are so. We claim that one can compute sets $U_1, U_2 \in \mathcal{C}$ such that $L_1 \cap L(\mathcal{A}_2) = L(\mathcal{A}, U_1)$ and $L_2 \cap L(\mathcal{A}_1) = L(\mathcal{A}, U_2)$. The set T of transitions of $\mathcal{A}$ is a subset of the product $T_1 \times T_2$ of transitions of $\mathcal{A}_1$ and $\mathcal{A}_2$, and thus there are obvious projections functions $\pi_1 : T \to T_1$ and $\pi_2 : T \to T_2$. If we enumerate the transition sets, say $T_1 = \{t_1^1, \ldots, t_1^m\}$, $T_2 = \{t_2^1, \ldots, t_2^n\}$, and $T = \{t_1, \ldots, t_\ell\}$ with $\ell \leq m \cdot n$, we obtain $\pi_1 : \{1, \ldots, \ell\} \to \{1, \ldots, m\}$ and $\pi_2 : \{1, \ldots, \ell\} \to \{1, \ldots, n\}$. We use these projections to define two linear (and in particular, affine) functions $\psi_1 : \mathbb{N}^\ell \to \mathbb{N}^m$ and $\psi_2 : \mathbb{N}^\ell \to \mathbb{N}^n$ which instead of counting transitions in T, count the corresponding transitions in T_1 or in T_2, respectively; formally,

$$\psi_1(v)[j] = \sum_{i:\pi_1(i)=j} v[i] \qquad \psi_2(v)[j] = \sum_{i:\pi_2(i)=j} v[i].$$

Finally, we set $U_1 := \psi_1^{-1}(S_1)$ and $U_2 := \psi_2^{-1}(S_2)$. Intuitively, U_1 and U_2 are as S_1 and S_2, except that instead of single transitions of $\mathcal{A}_1$ or $\mathcal{A}_2$ they are seeing pairs of transitions, and simply ignore one of them. Since $\mathcal{C}$ is closed under inverse images of affine mappings by assumption, $U_1, U_2 \in \mathcal{C}$. For the rest of the proof we may thus assume that the input automata are $(\mathcal{A}, U_1)$ and $(\mathcal{A}, U_2)$.

Step 2: Regular partitioning using skeletons. We finitely partition Σ^* s.t. words belonging to the same partition behave similarly with respect to automaton $\mathcal{A}$. We use the notion of *skeleton* of a run, defined already in [3], where it was used to solve the regularity problem

of *unambiguous* Parikh automata. The idea is to traverse a run from left to right while removing (and counting) simple cycles visiting states that have already appeared.

A *simple cycle* is a sequence of transitions $c = t_1 \dots t_n \in T^*$, where $t_i = (q_{i-1}, a_i, q_i)$, starting and ending in the same state $q_0 = q_n$ where $q_1, \dots, q_n$ are pairwise distinct. Two simple cycles c, d are *equivalent* if one is a cyclic permutation of the other. Let $[c]$ denote the equivalence class of c, and let $[c_1], \dots, [c_m]$ be a fixed enumeration of all such equivalence classes. (Since a simple cycle cannot visit the same state twice, except the initial state, it has length at most n, and thus the number of simple cycles, and also of equivalence classes thereof, is $m \leq d^n$, where d is the number of transitions of the automaton.) The skeleton is an inductively defined function from runs to pairs consisting of a run and a vector $v \in \mathbb{N}^m$. In the base case, $\textsc{skel}(\varepsilon) = (\varepsilon, 0)$. For the induction step, suppose that $\textsc{skel}(t_1 \dots t_{k-1}) = (u_1 \dots u_\ell, v)$ is already defined, and let q be the ending state of the new transition t_k. If q does not appear in the run $u_1 \dots u_\ell$, then we put $\textsc{skel}(t_1 \dots t_k) = (u_1 \dots u_\ell t_k, v)$. Otherwise, let u_h, for $h < \ell$, be the last transition that ends in state q, and consider the cycle $c = u_{h+1} \dots u_\ell t_k \in [c_j]$ (for some $1 \leq j \leq m$). We have two cases to consider. If all states visited by this cycle appeared before in $u_1 \dots u_h$, then we call this cycle *absorbed* and we remove it by putting $\textsc{skel}(t_1 \dots t_k) = (u_1 \dots u_h, v + e_j)$, where e_j is the vector which is 1 in coordinate j, and 0 everywhere else. Otherwise, we just put $\textsc{skel}(t_1 \dots t_k) = (u_1 \dots u_\ell t_k, v)$.

We remove only simple cycles visiting states that have already appeared before in order to have the following useful property.

▶ **Proposition 2.** *If* $\textsc{skel}(\rho) = (\hat{\rho}, v)$*, then* ρ *and* $\hat{\rho}$ *visit the same set of states.*

By abusing nomenclature, we call a run ρ a *skeleton* if $\textsc{skel}(\rho) = (\rho, v)$, for some $v \in \mathbb{N}^m$. It is easy to see that the length of a skeleton is at most n^2, where n is the number of states in the automaton $\mathcal{A}$. (Assume towards a contradiction that the length of the skeleton is longer than n^2. By the pigeonhole principle, some state is thus visited more than n times, so there are at least n cycles in between two consecutive occurrences of this state in the skeleton. Therefore it is impossible that each loop contains some new state not present in all the previous loops, and thus one of these loops should be removed during the process of creating the skeleton, a contradiction.) Consequently, if d is the total number of transitions of $\mathcal{A}$, then there are at most d^{n^2} skeletons. Let $\rho_1, \dots, \rho_h$ be all the skeletons, with $h \leq d^{n^2}$. We define K_i to be the set of all words w having an accepting run ρ in automaton $\mathcal{A}$ with $\textsc{skel}(\rho) = \rho_i$. Since $\mathcal{A}$ is deterministic we know that $K_i \cap K_j = \emptyset$ for $i \neq j$. Therefore $K_1, \dots, K_h$ and $K_{h+1} = \Sigma^* \setminus (\bigcup_{1 \leq i \leq h} K_i)$ form a partition of Σ^*. All languages K_i are necessarily regular, since the skeleton can be computed by a finite automaton. The following lemma can be seen as a generalization of Proposition 1 and it is immediate to prove.

▶ **Lemma 11.** *Let* Σ^* *be partitioned into regular languages* $K_1, \dots, K_k$*. Two languages* $L_1, L_2 \subseteq \Sigma^*$ *are regular separable if, and only if,* $L_1 \cap K_i$ *and* $L_2 \cap K_i$ *are regular separable for every* $i \in \{1, \dots, k\}$*.*

It remains to decide regular separability for the languages $L(\mathcal{A}, U_1) \cap K_i$ and $L(\mathcal{A}, U_2) \cap K_i$. In the following, fix a skeleton ρ and the set of words K with skeleton ρ. Since we have fixed a skeleton, we assume w.l.o.g. that the acceptance conditions U_1, U_2 are included in $\Pi(K)$.

Step 3: Reduction to unary separability for $\mathcal{C}$. Let the set of transitions of the automaton be $T = \{t_1, \dots, t_d\}$ (thus $\rho \in T^*$), and let $\mu : \mathbb{N}^m \to \mathbb{N}^d$ be the following affine function that

transforms counting cycles into counting transitions:

$$\mu(x_1, \ldots, x_m) = \Pi(\rho) + \sum_{1 \leq j \leq m} \Pi([c_j]) \cdot x_j.$$

Since $\Pi(c) = \Pi(d)$ for $c, d \in [c_j]$, $\Pi([c_j])$ is well-defined. Notice that μ is affine, and not linear, since we must take into account the initial cost of the skeleton $\Pi(\rho)$. Let $V_1 = \mu^{-1}(U_1)$ and $V_2 = \mu^{-1}(U_2)$ be the corresponding sets counting cycles instead of transitions. Since we assumed $U_1, U_2 \subseteq \Pi(K)$, every vector $v \in V_1$ is realizable by an accepting run $\hat{\rho}$ s.t. $\textsc{skel}(\hat{\rho}) = (\rho, v)$. Since $\mathcal{C}$ is closed under the inverse image of affine mappings, $V_1, V_2 \in \mathcal{C}$.

▶ **Lemma 12.** *The following two conditions are equivalent:*
1. *The two languages $L(\mathcal{A}, U_1) \cap K, L(\mathcal{A}, U_2) \cap K \subseteq \Sigma^*$ are regular separable.*
2. *The two sets of vectors $V_1, V_2 \subseteq \mathbb{N}^m$ are unary separable.*

Proof. For the implication 1) $\Rightarrow$ 2), suppose R is a regular language separating $L(\mathcal{A}, U_1) \cap K$ and $L(\mathcal{A}, U_2) \cap K$. For two words $x, y \in \Sigma^*$, define $x \equiv_R y$ if $x \in R \iff y \in R$. Fix $\omega \in \mathbb{N}$ such that for all words $x, y, z \in \Sigma^*$,

$$xy^{\omega}z \;\equiv_R\; xy^{2\omega}z. \tag{1}$$

It is easy to see that for every regular language R such ω exists. The simplest way of showing this is to consider the syntactic monoid M of R and to let ω be its idempotent power, i.e., a number such that $m^{\omega} = (m^{\omega})^2$ for every $m \in M$.

Recall n-unary equivalence: $u \equiv_n v$ if $u[i] \equiv v[i] \mod n$ and moreover $u[i] \leq n \iff v[i] \leq n$ for every coordinate $1 \leq i \leq m$. It is enough to show that for all $v_1 \in V_1, v_2 \in V_2$ it holds $v_1 \not\equiv_{\omega} v_2$. Indeed, if this is the case, the unary set $S = \{v \in \mathbb{N}^m \mid \exists_{v_1 \in V_1} v \equiv_{\omega} v_1\}$ separates V_1 and V_2.

Suppose, towards a contradiction, that there are some $v_1 \in V_1, v_2 \in V_2$ such that $v_1 \equiv_{\omega} v_2$. There are runs ρ_1, ρ_2 s.t. $\textsc{skel}(\rho_1) = (\rho, v_1)$ and $\textsc{skel}(\rho_2) = (\rho, v_2)$. We extend the equivalence $\equiv_R$ on runs by saying that $\rho_1 \equiv_R \rho_2$ if their two labellings are $\equiv_R$-equivalent. Since the labelling of ρ_1 is in $L(\mathcal{A}, U_1) \cap K$, and similarly for ρ_2, if $\rho_1 \equiv_R \rho_2$, then we derive a contradiction since R was supposed to separate $L(\mathcal{A}, U_1) \cap K$ and $L(\mathcal{A}, U_2) \cap K$. While in general $\rho_1 \equiv_R \rho_2$ does not hold, we can construct two *canonical runs* $\hat{\rho}_1$ and $\hat{\rho}_2$ s.t. (1) $\textsc{skel}(\hat{\rho}_1) = (\rho, v_1)$ (thus the labelling of $\hat{\rho}_1$ is also in $L(\mathcal{A}, U_1) \cap K$), (2) $\textsc{skel}(\hat{\rho}_2) = (\rho, v_2)$ (similarly), and (3) $\hat{\rho}_1 \equiv_R \hat{\rho}_2$ (thus bringing the contradiction). We show the construction for $\hat{\rho}_1$; the one for $\hat{\rho}_2$ is similar. By Proposition 2, states visited by the run ρ_1 are among those visited by the skeleton ρ, and in particular every absorbed cycle (i.e., $v_1[j] \neq 0$) also has this property. While in general a simple cycle $d \in [c_j]$ starting (and ending) at some state q cannot be reintroduced in the skeleton ρ at any position labelled by q (because not all states in d need to have appeared before this position), there always is a position i_j in the skeleton labelled by some state q_j and a simple cycle $\hat{c}_j \in [c_j]$ *starting (and ending) at* q_j s.t. all states in $\hat{c}_j$ have appeared already before position i_j in the skeleton. Assume w.l.o.g. that $i_1 \leq \cdots \leq i_m$. It is possible to split the skeleton ρ as $\alpha_0 \cdots \alpha_m$ s.t., for every $0 \leq j \leq m$, $\alpha_j \in T^*$ is a sequence of transitions, and the prefix $\alpha_0 \cdots \alpha_{j-1}$ has length i_j (thus ends in q_j). Then, define $\hat{\rho}_1$ and similarly $\hat{\rho}_2$ as

$$\hat{\rho}_1 := \alpha_0 \hat{c}_1^{v_1[1]} \alpha_1 \cdots \alpha_{m-1} \hat{c}_m^{v_1[m]} \alpha_m \quad \text{and} \quad \hat{\rho}_2 := \alpha_0 \hat{c}_1^{v_2[1]} \alpha_1 \cdots \alpha_{m-1} \hat{c}_m^{v_2[m]} \alpha_m.$$

(This is well-defined since $v_1 \equiv_{\omega} v_2$ implies $v_1[j] \neq 0$ iff $v_2[j] \neq 0$.) Properties (1) and (2) above are guaranteed by construction. For Property (3), since $v_1 \equiv_{\omega} v_2$, by repetitive use of Equation (1) we have $\hat{\rho}_1 \equiv_R \hat{\rho}_2$. This concludes the proof of the first implication.

For proving the implication (2) ⇒ (1), suppose that a unary set S separates V_1 and V_2. We claim that the language $R = L(\mathcal{A}, \mu(S)) \cap K$ is regular and separates $L(\mathcal{A}, U_1) \cap K$ and $L(\mathcal{A}, U_2) \cap K$. We first verify that R separates these two languages. Clearly, $U_1 \subseteq \mu(V_1) \subseteq \mu(S)$, so $L(\mathcal{A}, U_1) \cap K \subseteq L(\mathcal{A}, \mu(S)) \cap K = R$. The disjointness of $L(\mathcal{A}, U_2) \cap K$ and R is shown by contradiction. Suppose that there is a word $w \in K$ belonging both to $L(\mathcal{A}, \mu(S))$ and to $L(\mathcal{A}, U_2)$, let ρ be the run of $\mathcal{A}$ over w and let $v = \Pi(\rho)$. We have $v \in \mu(S) \cap U_2$, which implies $v = \mu(s)$ for some $s \in S \cap \mu^{-1}(U_2) = S \cap V_2$. In consequence $S \cap V_2$ is nonempty, thus contradicting the assumption that S separates V_1 and V_2.

In order to prove that R is regular it suffices to prove that $L(\mathcal{A}, \mu(S))$ is regular. The finite nondeterministic automaton recognizing this language simulates a run $\rho = t_{i_1} \dots t_{i_\ell}$ of $\mathcal{A}$, and accepts when $\Pi(\rho) \in \mu(S)$. Since S is unary, the automaton can evaluate this condition using finite memory. For every cycle c_j, the automaton stores a vector $x_j < \Pi(c_j)$, and a number n_j up to the unary equivalence $\equiv_n$, with the following meaning: the vector $\Pi(c_i)$ has been already executed n_j times, and x_i is the current "remainder". Additionally, the automaton stores a vector $x \leq \Pi(\rho)$ which is counting those transitions on the skeleton which have not been counted as cycles. At every input letter the automaton guesses nondeterministically one of cycles c_i or the skeleton and updates x_j, n_j and x accordingly. The automaton accepts when $x = \Pi(\rho_i)$, $x_j = 0$ for all j, and $(n_1, \dots, n_m) \in S$. ◀

6 Applications

We derive two corollaries of Theorem 1. By a *projection* we mean a function $\pi_{k,I} : \mathbb{N}^k \to \mathbb{N}^{|I|}$, for $I \subseteq \{1 \dots k\}$, that drops coordinates not in I. We start with a simple but useful lemma:

▶ **Lemma 13.** *If a class $\mathcal{C} \subseteq \bigcup_{d \in \mathbb{N}} \mathcal{P}(\mathbb{N}^d)$ contains all semilinear sets and is effectively closed under intersections, projections, and inverse images of projections, then it is effectively closed under inverse images of affine maps.*

Proof. Let S be a set in $\mathcal{C}$ and $f : \mathbb{N}^k \to \mathbb{N}^\ell$ be an affine map defined by $f(u) = Mu + v$ for $M = (m_{i,j})$ a matrix of dimension $\ell \times k$ and v a vector of dimension ℓ. Let $e_j \in \mathbb{N}^k$ be the vector s.t. $e_j[j] = 1$ and 0 otherwise, and let $m_j = (m_{1,j}, m_{2,j}, \dots, m_{\ell,j})$ be the (transpose of) the j-th column of M. First remark that the set $E_1 = \{(x, f(x)) \mid x \in \mathbb{N}^k\} \subseteq \mathbb{N}^{k+\ell}$ is linear with base $(0^k, v)$ and periods $\{p_1, \dots, p_k\}$, where $p_j = (e_j, m_j) \in \mathbb{N}^{k+\ell}$. Thus, $E_1 \in \mathcal{C}$. Therefore the set $E_2 = E_1 \cap \pi^{-1}_{k+\ell,I}(S)$ is also in $\mathcal{C}$, for $I = \{k+1, \dots, k+\ell\}$. Finally, we conclude since $\pi_{k+\ell,J}(E_2) = f^{-1}(S)$ with $J = \{1, \dots, k\}$. ◀

▶ **Corollary 14.** *Regular separability is decidable for nondeterministic Parikh automata.*

Proof. In order to apply Theorem 1 for $\mathcal{C}$ being semilinear sets, we need to know that the class of semilinear sets is robust. First, Lemma 13 yields effective closure under inverse images of affine maps, as semilinear sets are effectively closed under boolean combinations, images, and inverse images of projections. Second, decidability of the unary separability problem for semilinear sets is a corollary of the main result in [4]. This theorem states that separability of rational relations in $\Sigma^* \times \mathbb{N}^m$ by recognizable relations is decidable. If we ignore the Σ^* component we get the same result for rational and recognizable relations in $\mathbb{N}^m$, which are exactly semilinear sets and unary sets, respectively. ◀

For the second corollary we have to introduce vector addition systems and sections thereof. A d-dimensional *vector addition system* (VAS) is a pair $V = (s, T)$, where $s \in \mathbb{N}^d$ is a *source* configuration and $T \subseteq_{\text{FIN}} \mathbb{Z}^d$ is a finite set of *transitions*. A *run* is a sequence

$$(v_0, t_0, v_1), (v_1, t_1, v_2), \dots, (v_{n-1}, t_{n-1}, v_n) \in \mathbb{N}^d \times T \times \mathbb{N}^d$$

such that for all $i \in \{0, \ldots, n-1\}$ we have $v_i + t_i = v_{i+1}$ and $v_0 = s$. The *target* of this run is the configuration v_n. The *reachability set* of a VAS V is the set of targets of all its runs.

In order to ensure robustness, we slightly enlarge the family of VAS reachability sets to *sections* thereof. The intuition about a section is that we fix values on a subset of coordinates in vectors, and collect all the values that can occur on the other coordinates. For a subset $I \subseteq \{1, \ldots, d\}$, the projection $\pi_{d,I}$ extends element-wise to sets of vectors $S \subseteq \mathbb{N}^d$, denoted $\pi_{d,I}(S)$. For a vector $u \in \mathbb{N}^{d-|I|}$, the *section* of S w.r.t. I and u is the set

$$\pi_{d,I}(\{v \in S \mid \pi_{d,\{1,\ldots,d\}\setminus I}(v) = u\}) \subseteq \mathbb{N}^{|I|} \ .$$

We denote by $\mathcal{C}_{\text{SEC-VAS}}$ the family of all sections of VAS reachability sets.

▶ **Corollary 15.** *$\mathcal{C}_{\text{SEC-VAS}}$-Parikh automata.*

Proof. We apply Theorem 1 for $\mathcal{C} = \mathcal{C}_{\text{SEC-VAS}}$; we thus need to show that class $\mathcal{C}_{\text{SEC-VAS}}$ is robust. Decidability of unary separability of sets from $\mathcal{C}_{\text{SEC-VAS}}$ is shown in Theorem 9 in [5]. Effective closure of $\mathcal{C}$ under inverse images of affine functions will follow by Lemma 13 once we prove all its assumptions. First, $\mathcal{C}_{\text{SEC-VAS}}$ contains all semilinear sets. Effective closure under intersections is shown in Proposition 7 in [5]. Effective closure under inverse images of projections is easy: extend the VAS with additional coordinates, and allow it to arbitrarily increase these coordinates. Finally, to see that $\mathcal{C}_{\text{SEC-VAS}}$ is effectively closed under projections consider a section $S \subseteq \mathbb{N}^d$ of the reachability set of a VAS V, and a subset of coordinates $I \subseteq \{1, \ldots, d\}$. We construct a VAS V' which is like V, but additionally allows to decrease every coordinate from $\{1, \ldots, d\} \setminus I$. The projection $\pi_{d,I}(S)$ of S onto I is a section of the reachability set of V' defined similarly as S, but with an additional requirement that all coordinates from $\{1, \ldots, d\} \setminus I$ have value 0. ◀

7 Conclusions

We have shown that the regular separability problem for $\mathcal{C}$-Parikh automata is decidable, for every class $\mathcal{C}$ of acceptance conditions satisfying mild assumptions. In particular, we have shown decidability for $\mathcal{C}$ being either the semilinear sets or sections of VAS reachability sets. We have complemented our positive result by proving undecidability for visibly one counter languages, which sharpens two existing undecidability results.

The complexity of our algorithm depends on two factors: the complexity of unary separability for $\mathcal{C}$, and the complexity of computing inverse images of sets in $\mathcal{C}$ under affine mappings. The first factor is the dominant one here, since computing inverse images can be shown to be in PTIME for both semilinear sets and sections of VAS reachability sets (cf. [5]). Under this assumption, it can be seen that our reduction from nondeterministic to deterministic automata in Sec. 4 is also PTIME. Moreover, when reducing to unary separability in Sec. 5, since there are at most exponentially many skeletons and simple cycles, we obtain exponentially many unary separability instances, each of exponential size. Therefore, all together our reduction can be performed in exponential space.

On the other hand, the complexity of unary separability for semilinear sets and VAS sections is much higher. For semilinear sets, no upper bounds have been published for this problem, but an analysis of the algorithm of [4] yields an elementary bound[4]. For VAS sections, one can easily see that unary separability is at least as hard as the VAS reachability problem [5], which is hard for exponential space and not known to be primitive recursive.

[4] The problem becomes PTIME for the subclass of *diagonal* linear sets, in which case unary separability becomes the same as *modular separability*, and the latter problem is PTIME [4].

References

1 Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):16:1–16:43, May 2009. doi:10.1145/1516512.1516518.

2 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. On the Expressiveness of Parikh Automata and Related Models. In *Proc. of NCMA'11*, pages 103–119, 2011.

3 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Unambiguous constrained automata. *Int. J. Found. Comput. Sci.*, 24(7):1099–1116, 2013. doi:10.1142/S0129054113400339.

4 Christian Choffrut and Serge Grigorieff. Separability of rational relations in $A^* \times \mathbb{N}^m$ by recognizable relations is decidable. *Inf. Process. Lett.*, 99(1):27–32, 2006.

5 Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. Separability of Reachability Sets of Vector Addition Systems. In *Proc. of STACS'17*, volume 66 of *LIPICs*, pages 24:1–24:14, 2017. doi:10.4230/LIPIcs.STACS.2017.24.

6 Wojciech Czerwinski and Slawomir Lasota. Regular separability of one counter automata. In *Proc. of LICS'17*, 2017. To appear.

7 Wojciech Czerwiński, Wim Martens, and Tomás Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Proc. of ICALP'13*, pages 150–161, 2013.

8 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, and Marc Zeitoun. A note on decidable separability by piecewise testable languages. In *Proc. of FCT'15*, pages 173–185, 2015.

9 Jean Goubault-Larrecq and Sylvain Schmitz. Deciding piecewise testable separability for regular tree languages. In *Proc. of ICALP'16*, pages 97:1–97:15, 2016. doi:10.4230/LIPIcs.ICALP.2016.97.

10 Harry B. Hunt III. On the decidability of grammar problems. *J. ACM*, 29(2):429–447, 1982.

11 Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978.

12 Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In *Proc. of ICALP'03*, pages 681–696, 2003. doi:10.1007/3-540-45061-0_54.

13 Eryk Kopczynski. Invisible pushdown languages. In *Proc. of LICS'16*, pages 867–872, 2016. doi:10.1145/2933575.2933579.

14 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. In *Proc. of FSTTCS'13*, pages 363–375, 2013.

15 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In *Proc. of MFCS'13*, pages 729–740, 2013.

16 Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In *Proc. of ICALP'14*, pages 342–353, 2014.

17 Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Log. Methods Comput. Sci.*, 12(1), 2016.

18 Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2):231–250, 1976.

An Efficient Algorithm to Decide Periodicity of b-Recognisable Sets Using MSDF Convention*

Bernard Boigelot[1], Isabelle Mainz[2], Victor Marsault[†3], and Michel Rigo[4]

1 **Montefiore Institute & Department of Mathematics, Université de Liège, Liège, Belgium**
bernard.boigelot@ulg.ac.be
2 **Montefiore Institute & Department of Mathematics, Université de Liège, Liège, Belgium**
isabelle.mainz@ulg.ac.be
3 **Montefiore Institute & Department of Mathematics, Université de Liège, Liège, Belgium**
victor.marsault@ulg.ac.be
4 **Montefiore Institute & Department of Mathematics, Université de Liège, Liège, Belgium**
m.rigo@ulg.ac.be

Abstract

Given an integer base $b > 1$, a set of integers is represented in base b by a language over $\{0, 1, ..., b-1\}$. The set is said to be b-recognisable if its representation is a regular language. It is known that eventually periodic sets are b-recognisable in every base b, and Cobham's theorem implies the converse: no other set is b-recognisable in every base b.

We are interested in deciding whether a b-recognisable set of integers (given as a finite automaton) is eventually periodic. Honkala showed that this problem is decidable in 1986 and recent developments give efficient decision algorithms. However, they only work when the integers are written with the least significant digit first.

In this work, we consider the natural order of digits (Most Significant Digit First) and give a quasi-linear algorithm to solve the problem in this case.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases integer-base systems, automata, recognisable sets, periodic sets

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.118

1 Introduction

Let $b > 1$ be an integer base. We let $[\![b]\!] = \{0, 1, \ldots, b-1\}$ denote the canonical alphabet of base-b digits. If $u = u_\ell \cdots u_0$ belongs to $[\![b]\!]^*$, we let $\overline{u}$ denote the *value* of u in base b, *i.e.*, $\overline{u} = \sum_{i=0}^{\ell} u_i\, b^i$. Note that the leftmost digit is the most significant one. We let $\langle n \rangle$ denote the (shortest) *base-b representation* of n. We set $\langle 0 \rangle$ to be the empty word ε. If reference to the base b is needed, we write $\langle n \rangle_b$. Thus $\langle n \rangle$ is the unique word u over $[\![b]\!]$ not starting with

* An extended version of this paper is available in arXiv:1702.03715 [8], https://arxiv.org/abs/1702.03715.

† Corresponding author. Supported by a Marie Skłodowska-Curie fellowship, co-funded by the European Union and the University of Liège, Belgium.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 118; pp. 118:1–118:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

0 and such that $\overline{u} = n$. Moreover, for every $u \in [\![b]\!]^*$ such that $\overline{u} = n$, there exists $i \geq 0$ such that $u = 0^i\langle n\rangle$.

1.1 Our contribution

In this paper, we develop an algorithm to decide whether a given deterministic automaton $\mathcal{A}$ over the alphabet $[\![b]\!]$ accepts, by value, an (eventually) periodic set of integers. More precisely, the question is to decide whether there exist integers $p \geq 1$ and $N \geq 0$ such that, for all words $u \in [\![b]\!]^*$, if $\overline{u} \geq N$, then u is accepted by $\mathcal{A}$ if and only if $\langle \overline{u} + p\rangle$ is accepted as well. *Acceptance by value* means that words sharing the same value are either all accepted or all rejected. Stated otherwise, a word u is accepted by $\mathcal{A}$ if and if only if $0u$ is accepted. In the literature, one also finds the term "saturated language". The main result of this paper is the following.

▶ **Theorem 1.** *Given an integer base $b > 1$ and a n-state deterministic automaton $\mathcal{A}$ over the alphabet $[\![b]\!]$, it is decidable in $O(bn \log n)$ time whether or not $\mathcal{A}$ accepts, by value, some eventually periodic set of integers.*

Due to space constraints, this paper deals with the purely periodic case only. The general case is treated similarly [8].

Theorem 1 relies on a characterisation of the minimal automata accepting by value purely periodic sets. This characterisation is given Section 4. It relies on the notion of ultimately-equivalent states which is previously introduced in Section 2.2. In Section 3, we study the structure of the naive automaton accepting an arbitrary purely periodic sets, and its minimisation.

We stress the fact that the input automaton $\mathcal{A}$ reads words most significant digit first (MSDF). This is an important difference with other results discussed in the literature. For instance, an efficient algorithm to solve this decision problem is provided for automata reading least significant digit first (LSDF) [15, 16]. One can therefore think that it is enough to take the reversal of $\mathcal{A}$ and thus consider entries LSDF. Nevertheless, the reversal of $\mathcal{A}$ has first to be determinised. This potentially leads to an exponential blow-up in the number of states and thus to an inefficient procedure. For instance, this event occurs for the language $L_n = 0^*\,1\,(0+1)^n\,1\,(0+1+\varepsilon)^n$ and its mirror K_n: the number of states in the minimal automaton accepting L_n (resp. K_n) grows linearly (resp. exponentially) with n. Evaluating L_n as MSDF encodings or K_n as LSDF encodings yields the same finite (thus eventually periodic) set of integers.

1.2 Motivations and related results

We say that a set $X \subseteq \mathbb{N}$ is *b-recognisable* if $\langle X\rangle_b$ is accepted by some finite automaton. One reason why eventually periodic sets of integers play a special role comes from the celebrated theorem of Cobham about the dependence to the base of b-recognisability.

▶ **Theorem** (Cobham, [12]). *Let $b, c > 1$ be two multiplicatively independent integers. A set X of integers is such that the languages $\langle X\rangle_b$ and $\langle X\rangle_c$ are both accepted by finite automata if and only if X is eventually periodic.*

In combinatorics on words, when studying morphic words (for details and definitions, for instance, see [2, 5]), Cobham's theorem can be reformulated as follows. Let $b, c > 1$ be two multiplicatively independent integers. An infinite word $\mathbf{x}$ is both b-automatic and c-automatic if and only if $\mathbf{x}$ is of the form uv^ω where u, v are finite words. Indeed, a set

of integers is b-recognisable if and only if its characteristic sequence is b-automatic. The decision problem considered in our Theorem 1 is well known to be decidable.

▶ **Theorem** (Honkala, [14]). *It is decidable whether or not a given b-automatic word is eventually periodic.*

Complexity issues are however not considered at all in Honkala's paper. The decidability of our problem of interest can also be obtained using a first-order logic characterisation of b-recognisable sets given by Büchi's theorem, and the fact that Presburger arithmetic is decidable [10, 1]. These independent approaches all lead to decision procedures with exponential complexity.

Using LSDF convention, efficient decision procedures are known. First, Leroux obtained a quadratic decision procedure [15] for eventually-periodic b-recognisable sets of integers, that relies on intricate geometrical constructions. (Leroux's result is indeed stated in a multi-dimensional setting, *i.e.*, the problem is to decide whether or not a b-recognisable subset of $\mathbb{N}^d$ is semi-linear.) Still using LSDF convention, the third author and Sakarovitch designed a quasilinear algorithm [16]. The general idea is similar to the one we use here: finding characteristic properties that are preserved by minimisation. However, the criterion used in the present work is mostly unrelated to [16] and yields a very different decision algorithm.

1.3 Generalisation to real numbers

Real numbers can be encoded in a base $b > 1$ by extending positional encoding to infinite words: A word encoding a real is composed of a finite prefix corresponding to an integer part, followed by a single occurrence of a distinguished symbol acting as a separator, and an infinite suffix representing a fractional part. Infinite-word automata are then able to recognise sets of reals. It has been established that *weak deterministic automata*, a restricted class of infinite-word automata, are sufficiently expressive for recognising all sets definable in mixed integer and real first-order additive arithmetic [7].

The properties of sets of real numbers that can be recognised by weak deterministic automata in all bases $b > 1$ have been investigated [6]. Such sets generalise to the real domain the notion of eventual periodicity; they precisely correspond to finite combinations of eventually periodic sets of integers, and intervals of $[0, 1]$. Checking whether an automaton recognises such a set can be done by first splitting this automaton into finite-state machines operating on the integer and fractional parts of encodings. The former are then checked in the same way as for MSDF integer encodings, and the latter by verifying that they obey the structure documented in [6].

1.4 Generalisation to other numeration systems

Automatic words form a particular class of morphic words. Similarly, integer-base systems are special cases of more general numeration systems such as those built on a linear recurrent sequence. One can define a *numeration system* as a one-to-one map s from $\mathbb{N}$ to a language L over a finite alphabet. The integer n is mapped to its representation $s(n)$ within the considered system. Hence, it is natural to ask, for given a numeration system s and a subset M of L accepted by a finite automaton $\mathcal{A}$, whether or not the s-recognisable set $s^{-1}(M) \subseteq \mathbb{N}$ is eventually periodic.

On the one hand, Honkala's result is extended as follows. It is decidable whether or not a given morphic word is eventually periodic [13, 17]. On the other hand, Büchi's theorem can

be extended to linear numeration systems whose characteristic polynomial is the minimal polynomial of a Pisot number. See, for details, [9]. In that setting, several decision problems in combinatorics on words, including the ultimate periodicity problem, are decidable [11]. Using Honkala's techniques, the decision problem considered in our Theorem 1 is generalised to a large class of numeration systems in [4]. In particular, there are systems in this class for which the logical setting may not be applied. For all these decidability results presented in a wider context, no efficient procedure is known.

2 Preliminaries

In this paper, we only consider deterministic accessible finite automata with an input alphabet of the form $[\![b]\!]$. We use the acceptance-by-value convention. Thus, we may assume that the initial state bears a loop with label 0. In particular, this will always be the case after minimisation. Let $\mathcal{A}$ be an automaton. Its set of states (resp. its initial state, its set of final states) is denoted by $Q_{\mathcal{A}}$ (resp. $i_{\mathcal{A}}$, $F_{\mathcal{A}}$). If the considered automaton is clear from the context, $(s \cdot u)$ is the state s' such that $s \xrightarrow{u} s'$. The language accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$. In this section, we recap basic results about automata.

2.1 Automaton morphisms and pseudo-morphisms

▶ **Definition 2.** Given two (accessible) automata $\mathcal{A}$ and $\mathcal{M}$ over $[\![b]\!]$, an *automaton morphism* $\mathcal{A} \to \mathcal{M}$ is a function $\phi : Q_{\mathcal{A}} \to Q_{\mathcal{M}}$ that satisfies:

$$\phi(i_{\mathcal{A}}) = i_{\mathcal{M}} \tag{1}$$

$$\forall s \in Q_{\mathcal{A}},\ \forall a \in [\![b]\!] \quad (s \cdot a) \text{ exists in } \mathcal{A} \iff (\phi(s) \cdot a) \text{ exists in } \mathcal{M} \tag{2}$$

$$\forall s, s' \in Q_{\mathcal{A}},\ \forall a \in [\![b]\!] \quad s \xrightarrow{a} s' \text{ in } \mathcal{A} \implies \phi(s) \xrightarrow{a} \phi(s') \text{ in } \mathcal{M} \tag{3}$$

$$F_{\mathcal{A}} = \phi^{-1}(F_{\mathcal{M}}) \tag{4}$$

▶ **Definition 3.** If a function ϕ satisfies (1), (2) and (3) but not necessarily (4), then we say that we have an *automaton pseudo-morphism*.

▶ **Definition 4.** Two states s, s' of an automaton $\mathcal{A}$ are *Nerode-equivalent* if, for every word u, $(s \cdot u)$ exists and is final if and only if $(s' \cdot u)$ exists and is final.

We recall the following classical result. See, for instance, [18].

▶ **Theorem 5** (Myhill–Nerode)**.** *Let $\mathcal{A}$ be a complete automaton. Among all the complete automata accepting $L(\mathcal{A})$, up to isomorphism, there exists a unique one with a minimal number of states, called the* minimisation of $\mathcal{A}$*. Moreover, if $\mathcal{M}$ denotes the minimisation of $\mathcal{A}$, then there exists an automaton morphism $\phi : \mathcal{A} \to \mathcal{M}$ (called the* minimisation morphism*) such that*

$$\forall s, s' \in \mathcal{A} \quad \phi(s) = \phi(s') \iff s \text{ and } s' \text{ are Nerode-equivalent.} \tag{5}$$

If $\mathcal{A}$ is an automaton and u is a word, we write $(\mathcal{A} \cdot u)$ as a shorthand for $(i_{\mathcal{A}} \cdot u)$, i.e., the state reached by the run of u in $\mathcal{A}$.

▶ **Lemma 6.** *Let $\mathcal{A}$ and $\mathcal{M}$ be two complete (and accessible) automata. There exists a pseudo-morphism $\mathcal{A} \to \mathcal{M}$ if and only if every pair of words u, u' such that $(\mathcal{M} \cdot u) \neq (\mathcal{M} \cdot u')$ also satisfies $(\mathcal{A} \cdot u) \neq (\mathcal{A} \cdot u')$.*

Proof. Forward direction. Since a pseudo-morphism ϕ respects transitions and the initial state, it follows that, for every word u, $(\mathcal{M}\cdot u) = \phi(\mathcal{A}\cdot u)$. The statement follows immediately.

Backward direction. For every state s, we choose a word u_s such that $(\mathcal{A}\cdot u_s) = s$ (such a word exists because $\mathcal{A}$ is accessible). We define a function $\phi : Q_{\mathcal{A}} \to Q_{\mathcal{M}}$ as follows. For every state $s \in Q_{\mathcal{A}}$, $\phi(s) = (\mathcal{M}\cdot u_s)$. Let us show that ϕ is an automaton pseudo-morphism.

Let s be a state of $\mathcal{A}$ and let u be a word such that $(\mathcal{A}\cdot u) = s$. Since $(\mathcal{A}\cdot u) = (\mathcal{A}\cdot u_s)$, the hypothesis implies $(\mathcal{M}\cdot u) = (\mathcal{M}\cdot u_s)$. The definition of ϕ is therefore independent of the choice of the words u_s.

In particular, $\phi(i_{\mathcal{A}}) = (\mathcal{M}\cdot u_{i_{\mathcal{A}}}) = (\mathcal{M}\cdot\varepsilon) = i_{\mathcal{M}}$ hence ϕ satisfies (1). Moreover, since both $\mathcal{A}$ and $\mathcal{M}$ are complete, and since ϕ is a total function, ϕ also satisfies (2). Let $t \xrightarrow{a} t'$ be a transition of $\mathcal{A}$. By definition $\phi(t) = (\mathcal{M}\cdot u_t)$ and since the definition of ϕ does not depend on the choice of the words u_s, we may assume that $u_{t'} = u_t a$. It then follows that

$$\phi(t') = (\mathcal{M}\cdot(u_t a)) = ((\mathcal{M}\cdot u_t)\cdot a) = \phi(t)\cdot a\ .$$

In other words, $\phi(t) \xrightarrow{a} \phi(t')$ is a transition of $\mathcal{M}$. ◀

2.2 Ultimately-equivalent states

Our decision procedure involves the determination of ultimately-equivalent states defined as follows.

▶ **Definition 7.** Let $\mathcal{A}$ be an automaton over $[\![b]\!]$. Let $m \geq 1$ be an integer. Two states s, s' of $\mathcal{A}$ are *m-ultimately-equivalent* if

$$\forall u \in [\![b]\!]^* \quad |u| \geq m \implies (s\cdot u) = (s'\cdot u)\ .$$

Two states are *ultimately-equivalent* if they are m-ultimately-equivalent for some $m \geq 1$.

▶ Remark. Note that ultimate-equivalence is indeed an equivalence relation: if s and s' are m-ultimately-equivalent while s' and s'' are m'-ultimately-equivalent, then s and s'' are $\max(m, m')$-ultimately-equivalent.

Given an automaton $\mathcal{A}$ over $[\![b]\!]$, the computation of this relation is easy. Let us build a directed graph $\mathcal{G} = (V, E)$ as follows. The vertex-set is $V = Q_{\mathcal{A}} \times Q_{\mathcal{A}}$ and the edge set is:

$$\forall (s,t),(s',t') \in V,\ s \neq t$$
$$(s,t) \to (s',t') \text{ in } \mathcal{G} \iff \exists a \in [\![b]\!] \text{ such that } \mathcal{A} \text{ features } \begin{cases} s \xrightarrow{a} s' \\ t \xrightarrow{a} t' \end{cases}\ . \quad (6)$$

In particular, vertices of the form (s, s) never qualify for the above condition and thus have no outgoing edges. Observe that two distinct states s, t of $\mathcal{A}$ are ultimately-equivalent if and only if (s, t) may not reach in $\mathcal{G}$ a strongly connected component.

Computing the strongly connected components of a graph is done in linear time (e.g., with Tarjan's algorithm [19]). Hence, the set of the pairs of states of $\mathcal{A}$ that are ultimately-equivalent may be computed in time $O(bn^2)$. This complexity can be improved as follows.

▶ **Proposition 8** (Béal, Crochemore, [3]). *Let $\mathcal{A}$ be an automaton over $[\![b]\!]$ and n the number of states in $\mathcal{A}$. The ultimate-equivalence classes of $\mathcal{A}$ may be computed in time $O(bn\log n)$.*

Proof Sketch. We take verbatim the algorithm in [3]. One starts from the trivial partition and iteratively merges states. Each step of the algorithm consists in merging two states that are 1-ultimately-equivalent. The purpose of Béal and Crochemore was to show that starting with a so-called AFT automaton $\mathcal{A}$, the result is the minimisation of $\mathcal{A}$. Starting with any automaton $\mathcal{A}$, the resulting automaton is not necessarily minimal. However, one can observe that its states are precisely the ultimate-equivalence classes of $\mathcal{A}$. ◀

As a direct consequence of the definition of an automaton morphism, ultimate-equivalence commutes with automaton morphisms.

▶ **Lemma 9.** *Let $\mathcal{A}$ and $\mathcal{M}$ be two automata such that there is an automaton morphism $\phi : \mathcal{A} \to \mathcal{M}$. Let s and s' be two states of $\mathcal{A}$ that are ultimately-equivalent (w.r.t. $\mathcal{A}$), then $\phi(s)$ and $\phi(s')$ are also ultimately-equivalent (w.r.t. $\mathcal{M}$).*

3 Purely periodic b-recognisable sets

The content of this section is the following. Section 3.1 gives the definition and main properties of the "naive" automaton $\mathcal{A}_{(p,R)}$ accepting the purely periodic set $R + p\mathbb{N}$. Then, in Section 3.2 we study the relationship between ultimate equivalence and Nerode equivalence in $\mathcal{A}_{(p,R)}$. In Section 3.3, we show how to extract relevant information on the period p from the minimisation of $\mathcal{A}_{(p,R)}$.

▶ **Notation 10.** *Let $p > 0$ and $b > 1$ be two integers. Throughout this section, the quantities k, d, j, ψ are fixed as follows.*

- *Let k, d be the unique integers such that $p = k\,d$ where k is the greatest divisor of p coprime with b. In particular, the prime factors occurring in the prime decomposition of d all appear in the prime decomposition of b. Moreover, $(k, d) = 1$.*
- *Since $(k, b) = 1$, the order of b in $\mathbb{Z}/k\mathbb{Z}$ is well defined and denoted by ψ, i.e., $b^\psi \equiv 1\ [k]$.*
- *Let j be the least integer such that d is a divisor of b^j.*

Let $s < k$ and $t < d$ be two integers. Let $\langle s, t\rangle$ denote the integer of $\mathbb{Z}/p\mathbb{Z}$ congruent to s modulo k and t modulo d. This integer is unique by the Chinese remainder theorem. Note that if n is an integer less than p, then $n = \langle n\%k, n\%d\rangle$ where $n\%k$ denote the remainder of the division of n by k.

3.1 The automaton $\mathcal{A}_{(p,R)}$ and its minimisation

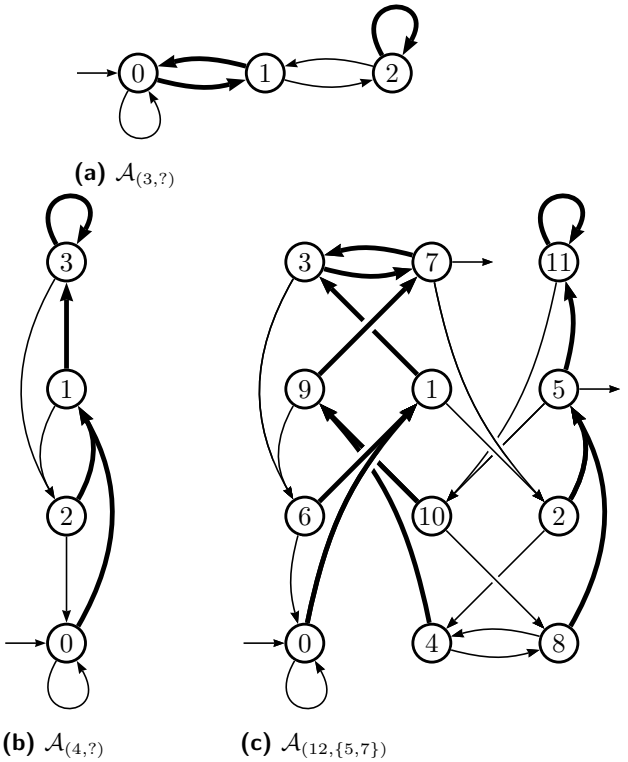
▶ **Definition 11.** A subset P of integers is *purely periodic*, if there exist $p \geq 1$ and a subset $R \subseteq \{0, \dots, p-1\}$ such that $P = R + p\mathbb{N}$.

For instance, $\{0, 1\} + 4\mathbb{N}$ is purely periodic but $\{4, 5\} + 4\mathbb{N}$ is not. Let $p \geq 1$ be an integer and R be a subset of $\{0, \dots, p-1\}$. We say that the parameter (p, R) is *proper*, if p is the smallest period of the purely periodic set $R + p\mathbb{N}$. For instance, $(4, \{0, 1\})$ is proper but $(4, \{0, 2\})$ is not because $\{0, 2\} + 4\mathbb{N} = \{0\} + 2\mathbb{N}$.

The following definition is ubiquitous when dealing with periodic sets of integers. It is an easy exercise to show that this automaton accepts base-b representations of integers whose remainder modulo p belongs to R.

▶ **Definition 12.** We let $\mathcal{A}_{(p,R)}$ denote the automaton $\mathcal{A}_{(p,R)} = \langle [\![b]\!], \mathbb{Z}/p\mathbb{Z}, \delta, 0, R\rangle$ where δ is defined as

$$\forall n \in \mathbb{Z}/p\mathbb{Z},\ \forall a \in [\![b]\!] \quad n \xrightarrow{a} nb + a\ .$$



(a) $\mathcal{A}_{(3,?)}$ **(b)** $\mathcal{A}_{(4,?)}$ **(c)** $\mathcal{A}_{(12,\{5,7\})}$

Figure 1 The automaton $\mathcal{A}_{(12,\{5,7\})}$, as the product automaton of $\mathcal{A}_{(4,?)}$ by $\mathcal{A}_{(3,?)}$.

When we are only interested in the transitions of the automaton $\mathcal{A}_{(p,R)}$, it is sometimes convenient to leave the set of final states unspecified. In that case, we write $\mathcal{A}_{(p,?)}$ for the automaton where the final/non-final status of the states is not set.

▶ **Example 13.** Figure 1c shows $\mathcal{A}_{(12,\{5,7\})}$ in base 2. Transitions with label 1 (resp. 0) are represented with bold (resp. thin) edges.

As can be seen, for instance, in Figure 2, the automaton $\mathcal{A}_{(p,R)}$ is not necessarily minimal.

▶ **Lemma 14.** *For every word* $u \in [\![b]\!]^*$, $(\mathcal{A}_{(p,R)} \cdot u) = (\overline{u}\%p) = \langle \overline{u}\%k, \overline{u}\%d \rangle$.

Proof. This follows directly from the definition of the transition function of $\mathcal{A}_{(p,?)}$. ◀

▶ **Property 15.** *The automaton* $\mathcal{A}_{(p,R)}$ *is strongly connected.*

Proof. Let n, m be two states. The state n is of the form $\langle i, i' \rangle$. Let u be a word satisfying

$$\overline{u} \equiv \langle k - i, 0 \rangle [p] \ , \quad |u| \geq j \quad \text{and} \quad |u| \equiv 0 [\psi] \ .$$

The last two conditions are easily satisfied by adding a suitable number of leading zeroes. Reading u from n leads to the initial state 0. Obviously, reading $\langle m \rangle$ from 0 leads to m. ◀

The next lemma states that the automaton $\mathcal{A}_{(p,?)}$ is the product automaton $\mathcal{A}_{(k,?)} \times \mathcal{A}_{(d,?)}$. This easily follows from the Chinese remainder theorem and Lemma 14.

▶ **Lemma 16.** *For all integers* $s, s' \in \mathbb{Z}/k\mathbb{Z}$, $t, t' \in \mathbb{Z}/d\mathbb{Z}$ *and every word* $u \in [\![b]\!]^*$,

$$\langle s, t \rangle \xrightarrow{u} \langle s', t' \rangle \text{ in } \mathcal{A}_{(p,?)} \iff \begin{cases} s \xrightarrow{u} s' \text{ in } \mathcal{A}_{(k,?)} \\ t \xrightarrow{u} t' \text{ in } \mathcal{A}_{(d,?)} \end{cases}$$

The fact that k is coprime with b implies the following result.

▶ **Lemma 17.** *With the definition introduced in Notation 10, the automaton $\mathcal{A}_{(k,?)}$ is a group automaton: each letter induces a permutation on the set of states.*

Proof. Since k is coprime with b, the function $f_0 : \mathbb{Z}/k\mathbb{Z} \to \mathbb{Z}/k\mathbb{Z}$ defined by $s \mapsto sb$ is a permutation of $\mathbb{Z}/k\mathbb{Z}$. Hence, so is the function f_a defined by $s \mapsto (sb + a)$, for every letter $a \in [\![b]\!]$. The action of a in $\mathcal{A}_{(k,?)}$ is exactly f_a, a permutation of the states. ◀

3.2 Nerode-equivalence and ultimate-equivalence in $\mathcal{A}_{(p,R)}$

Within the setting of Example 13 where rows (resp. columns) of the product automaton $\mathcal{A}_{(p,R)} \approx \mathcal{A}_{(d,?)} \times \mathcal{A}_{(k,?)}$ correspond to the equivalence classes modulo d (resp. modulo k), the forthcoming Proposition 19 shows that Nerode-equivalent states in $\mathcal{A}_{(p,R)}$ must belong to the same column. See, for instance, Figure 2. Then, we show that all states belonging to the same column are ultimately-equivalent.

▶ **Lemma 18.** *If (p, R) is proper, then for all distinct integers i and i', $0 \leq i, i' < k$, the states id and $i'd$ are not Nerode-equivalent.*

Proof. Since (p, R) is proper and $id \neq i'd$, there exists an integer m such that $(id + m) \in R + p\mathbb{N}$ and $(i'd + m) \notin R + p\mathbb{N}$.

We let u denote a word such that $\overline{u} = m$ and $|u| \equiv 0[\psi]$ (in other words, u is the word $\langle m \rangle$ padded with an appropriate number of 0's); it thus holds that $b^{|u|} \equiv 1\ [k]$. Reading the word u respectively from the states id and $i'd$ leads to the states:

$$id \cdot u = idb^{|u|} + m \quad \text{and} \quad i'd \cdot u = i'db^{|u|} + m\ .$$

The integer $(idb^{|u|} + m)$ is congruent to $(id + m)$ modulo k (since $b^{|u|} \equiv 1\ [k]$) as well as modulo d (since both are obviously congruent to m) hence modulo p. The same reasoning also applies to the second state, finally yielding:

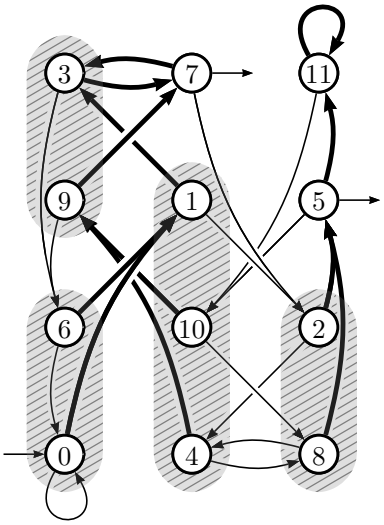
$$id \cdot u = id + m \quad \text{and} \quad i'd \cdot u = i'd + m\ .$$

The first state belongs to R and is thus final while the second does not belong to R and thus is not final. The word u is then a witness of the fact that id and $i'd$ are not Nerode-equivalent. ◀

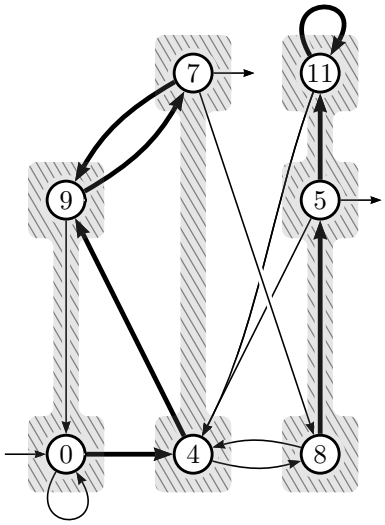
▶ **Proposition 19.** *Let (p, R) be proper. If i and i' are Nerode-equivalent states, then they are congruent modulo k.*

Proof. Proof by contrapositive. Let i and i' be two states that are not congruent modulo k. By definition of j, see Notation 10, the states $(i \cdot 0^j)$ and $(i' \cdot 0^j)$ are both congruent to 0 modulo d. However the operation $i \mapsto ib$ is a permutation of $\mathbb{Z}/k\mathbb{Z}$, hence $(i \cdot 0^j)$ and $(i' \cdot 0^j)$ are not congruent modulo k. It follows that $(i \cdot 0^j) = ld$ and $(i' \cdot 0^j) = l'd$ for some distinct $l, l' \in \mathbb{Z}/k\mathbb{Z}$. Lemma 18 then yields that these states are not Nerode-equivalent, hence that i and i' are not either. ◀

▶ **Lemma 20.** *Let s and s' be two states of $\mathcal{A}_{(p,R)}$. With the definition introduced in Notation 10, if $s \equiv s'[k]$, then s and s' are j-ultimately-equivalent.*



(a) Nerode-equivalence classes of $\mathcal{A}_{(12,\{5,7\})}$



(b) Pseudo-morphism equivalence classes in the minimisation of $\mathcal{A}_{(12,\{5,7\})}$

Figure 2 Minimisation morphism of $\mathcal{A}_{(12,\{5,7\})}$ and pseudo-morphism of its minimisation.

Proof. Let u be any word of length j. Since s and s' are congruent modulo k, there exists $i \in \mathbb{Z}/k\mathbb{Z}$ and $l, l' \in \mathbb{Z}/d\mathbb{Z}$ such that $s = \langle i, l\rangle$ and $s' = \langle i, l'\rangle$. Then, from Lemma 16 and using the fact that $lb^j \equiv 0\ [d]$, we get

$$(s \cdot u) = \langle ib^j + \overline{u},\ lb^j + \overline{u}\rangle = \langle ib^j + \overline{u},\ \overline{u}\rangle\ .$$

Similarly $(s' \cdot u) = \langle ib^j + \overline{u},\ \overline{u}\rangle = (s \cdot u)$. ◀

3.3 Circuits labelled by the digit 0

A circuit in which every arc is labelled by the digit 0 is called for short a *0-circuit*. For instance, the automaton $\mathcal{A}_{(12,\{5,7\})}$ depicted in Figure 1 has two such circuits: $0 \xrightarrow{0} 0$ and $4 \xrightarrow{0} 8 \xrightarrow{0} 4$. We will see that the number of states belonging to 0-circuits has a special meaning.

▶ **Lemma 21.** *A state of $\mathcal{A}_{(p,R)}$ is a multiple of d if and only if it belongs to a 0-circuit.*

Proof. Forward direction. It is enough to show that every state of the form id, for $i \in \mathbb{Z}/k\mathbb{Z}$, has a predecessor by 0 of the form $i'd$, $i' \in \mathbb{Z}/k\mathbb{Z}$. Simple arithmetic yields that $(b^{-1}i)d$ is suitable, where b^{-1} is the inverse of b in $\mathbb{Z}/k\mathbb{Z}$.

Backward direction. Proof by contrapositive. Let s be a state which is not a multiple of d. The state $(s \cdot 0^j)$ is a multiple of d. Therefore, for every integer $i \geq j$, the state $(s \cdot 0^i)$ is a multiple of d, hence is not equal to s. Since $\mathcal{A}_{(p,R)}$ is deterministic, $(s \cdot 0^i)$ cannot be equal to s for $i < j$ either. ◀

The next proposition follows from Lemmas 21 and 18. Recall that k is the largest integer coprime with b such that $p = k\,d$ and $d \geq 1$ (see Notation 10).

▶ **Proposition 22.** *If (p, R) is proper, the minimisation of $\mathcal{A}_{(p,R)}$ possesses exactly k states that belong to 0-circuits.*

4 Characterisation of automata accepting purely periodic sets

The next result will allow us to decide whether a deterministic automaton $\mathcal{A}$ over $[\![b]\!]$, given as input, is such that $\overline{L(\mathcal{A})}$ is a purely periodic set of integers, i.e., whether or not it is of the form $R + p\mathbb{N}$ for some R and p. We say that an equivalence relation $\sim_1$ is a *refinement* of another equivalence relation $\sim_2$ if for every x and y, $x \sim_1 y \implies x \sim_2 y$.

▶ **Theorem 23.** *Let $b > 1$ be a base and $\mathcal{A}$ a minimal automaton over $[\![b]\!]$. Let ℓ be the number of states in $\mathcal{A}$ that belong to 0-circuits. The automaton $\mathcal{A}$ accepts by value a purely periodic set of integers if and only if the following conditions are fulfilled.*

(a) *There exists a pseudo-morphism $\phi : \mathcal{A} \to \mathcal{A}_{(\ell,?)}$.*

(b) *The equivalence relation induced by ϕ is a refinement of the ultimate-equivalence relation.*

(c) *The initial state of $\mathcal{A}$ bears a self-loop labelled by the digit 0*

Proof of forward direction. Since the automaton $\mathcal{A}$ is minimal and accepts by value, the initial state of $\mathcal{A}$ necessarily bears a loop labelled by the digit 0; in other words, item (3) holds.

More precisely, since $\mathcal{A}$ accepts by value a purely periodic set of integers, there exists a smallest period p and a remainder-set $R \subseteq \{0, \dots, p-1\}$ such that $L(\mathcal{A}) = 0^*\langle R + p\mathbb{N}\rangle$. Note that (p, R) is proper by choice of p. We make use of Notation 10. In particular, k is the greatest divisor of p that is coprime with b.

Since $\mathcal{A}$ is minimal, it is isomorphic to the minimisation of any automaton accepting $L(\mathcal{A})$, in particular, to the minimisation of $\mathcal{A}_{(p,R)}$. It then follows from Proposition 22 that $\ell = k$.

To prove that there exists a pseudo-morphism $\phi : \mathcal{A} \to \mathcal{A}_{(k,?)}$, we will apply Lemma 6. Let u,u' be two words such that $(\mathcal{A}_{(k,?)} \cdot u) \neq (\mathcal{A}_{(k,?)} \cdot u')$. Let us show that $(\mathcal{A} \cdot u) \neq (\mathcal{A} \cdot u')$. Since $(\mathcal{A}_{(k,?)} \cdot u) \neq (\mathcal{A}_{(k,?)} \cdot u')$, we have that $\overline{u} \not\equiv \overline{u'}$ $[k]$. Due to Lemma 14, $(\mathcal{A}_{(p,R)} \cdot u)$ and $(\mathcal{A}_{(p,R)} \cdot u')$ are not congruent modulo k. It then follows from Proposition 19 that the states $(\mathcal{A}_{(p,R)} \cdot u)$ and $(\mathcal{A}_{(p,R)} \cdot u')$ are not Nerode-equivalent, which implies that $(\mathcal{A} \cdot u) \neq (\mathcal{A} \cdot u')$ because $\mathcal{A}$ is the minimisation of $\mathcal{A}_{(p,R)}$.

Let s and s' be two states of $\mathcal{A}$ such that $\phi(s) = \phi(s')$. We have to show that s and s' are ultimately-equivalent. Let u and u' be two words such $(\mathcal{A} \cdot u) = s$ and $(\mathcal{A} \cdot u') = s'$. Since ϕ is a pseudo-morphism, we get that

$$(\mathcal{A}_{(k,?)} \cdot u) = \phi(s) = \phi(s') = (\mathcal{A}_{(k,?)} \cdot u')$$

and so $\overline{u} \equiv \overline{u'}$ $[k]$. Applying Lemma 14 yields that the states $(\mathcal{A}_{(p,R)} \cdot u)$ and $(\mathcal{A}_{(p,R)} \cdot u')$ are congruent modulo k, and by Lemma 20, these states are ultimately-equivalent. Since $\mathcal{A}$ is the minimisation of $\mathcal{A}_{(p,R)}$, we have an automaton morphism $\mathcal{A}_{(p,R)} \to \mathcal{A}$. Finally, since ultimate-equivalence commutes with automaton morphism (Lemma 9), $(\mathcal{A} \cdot u) = s$ and $(\mathcal{A} \cdot u') = s'$ are ultimately-equivalent. ◀

Proof of backward direction. By assumption, for all $i \in \mathbb{Z}/\ell\mathbb{Z}$, the states in $\phi^{-1}(i)$ are ultimately-equivalent. For every integer $i \in \mathbb{Z}/\ell\mathbb{Z}$, we let m_i denote the least integer such that, for all s, s' in $\phi^{-1}(i)$, $(s \cdot u) = (s' \cdot u)$ whenever $|u| \geq m_i$. Let $m = \max\{m_i \mid i \in \mathbb{Z}/\ell\mathbb{Z}\}$.

Let u, u' be two words with respective values that are congruent modulo ℓb^m. Note that, in particular, $\overline{u}$ and $\overline{u'}$ are thus congruent modulo b^m. Let us show that u and u' reach the same state in $\mathcal{A}$.

Since $\mathcal{A}$ bears a self-loop labelled by 0 on the initial state, the word $0^m u$ is such that $\overline{0^m u} = \overline{u}$ and $\mathcal{A} \cdot 0^m u = \mathcal{A} \cdot u$. We may thus assume that u and u' are longer than m.

There exist factorisations $u = vw$ and $u' = v'w'$ such that the lengths of w and w' are both equal to m. Since $\overline{u}$ and $\overline{u'}$ are congruent modulo b^m, w and w' are equal: $u = vw$, $u' = v'w$.

Assume without loss of generality that $\overline{u} \geq \overline{u'}$. Hence $\overline{u} - \overline{u'} = (\overline{v} - \overline{v'})b^m$ is congruent to 0 modulo ℓb^m. We deduce that $\overline{v}$ and $\overline{v'}$ are congruent modulo ℓ. By Lemma 14, the respective runs of v and v' in $\mathcal{A}_{(\ell,?)}$ reach the same state: $(\mathcal{A}_{(\ell,?)} \cdot v) = (\mathcal{A}_{(\ell,?)} \cdot v')$. From assumption (1), we get $\phi(\mathcal{A} \cdot v) = \phi(\mathcal{A} \cdot v')$. In other words, the states $(\mathcal{A} \cdot v)$ and $(\mathcal{A} \cdot v')$ are ϕ-equivalent. Hence, by assumption (2), they are m_i-ultimately-equivalent. Since $|w| = m \geq m_i$ (by choice of m), we get that $(\mathcal{A} \cdot v \cdot w) = (\mathcal{A} \cdot v' \cdot w)$: the run in $\mathcal{A}$ of the words $u = vw$ and $u' = v'w$ indeed reach the same state.

We have just shown that words whose values are congruent modulo ℓb^m have runs in $\mathcal{A}$ reaching the same states, hence either all are accepted by $\mathcal{A}$ or none of them are. The run of a word u is then accepted by $\mathcal{A}$ if and only if $\langle \overline{u} \% (\ell b^m) \rangle$ is. Finally, a word u is accepted by $\mathcal{A}$ if and only if $\overline{u} \% (\ell b^m)$ belongs to the set $R \subseteq \{0, \ldots, \ell b^m - 1\}$, defined by

$$R = \{\ i \in \mathbb{Z}/\ell b^m \mathbb{Z} \mid (\mathcal{A} \cdot \langle i \rangle) \text{ is final } \}\ . \qquad \blacktriangleleft$$

▶ **Remark 24.** *In the proof of the forward direction, it was stated that $\ell = k$ (where k is the greatest divisor of the period which is coprime with the base). It is also the case in the backward direction. Indeed, the automaton $\mathcal{A}$ is shown to accept a purely periodic set of integers. Let (p, R) denotes the* ***proper*** *parameter of this set (it is not necessarily the one given in the proof). Since $\mathcal{A}$ is minimal, it is a quotient of $\mathcal{A}_{(p,R)}$. It then follows from Proposition 22 that, ℓ, the number of states belonging to 0-circuits, is equal to k, the greatest divisor of the period which is coprime with the base.*

4.1 Complexity and algorithmic issues

Theorem 23 yields an algorithm to decide whether a given deterministic automaton $\mathcal{A}$ accepts by value a purely periodic set of integers:

0. if necessary, minimise $\mathcal{A}$ and make it complete;
1. count the number ℓ of states of $\mathcal{A}$ that belong to 0-circuits;
2. build the automaton $\mathcal{A}_{(\ell,?)}$;
3. construct, if it exists, the pseudo morphism $\phi : \mathcal{A} \to \mathcal{A}_{(\ell,?)}$;
4. check whether, for all $x \in \mathbb{Z}/\ell\mathbb{Z}$, the states of $\phi^{-1}(x)$ are ultimately-equivalent.

Let us denote by n the number of states of $\mathcal{A}$. Step (0) can be carried out in $O(bn \log n)$ time. Steps (1) and (2) can obviously be performed in $O(bn)$ time. A morphism between deterministic automata, if it exists, can be computed by a single traversal of the bigger automaton; the same algorithm also works for pseudo-morphisms: Step (3) also runs in $O(bn)$ time. The ultimate-equivalence classes of $\mathcal{A}$ can be computed in time $O(bn \log n)$ from Proposition 8, hence so is the execution of Step (4).

▶ **Corollary 25.** *Let $b > 1$ be a base and $\mathcal{A}$ be a n-state deterministic automaton over $[\![b]\!]$. It is decidable in $O(bn \log n)$ time whether $\mathcal{A}$ accepts by value a purely periodic set of integers.*

▶ **Remark 26.** *Remark 24 gives a very fast rejection test. Indeed, before Step (2) we may check whether the integer ℓ (computed by Step (1)) is coprime with b. If it is not the case, $\mathcal{A}$ may be rejected already.*

▶ **Example 27.** We start with the minimal automaton $\mathcal{A}$ depicted in Figure 3. Step (1) is shown in Figure 4: $\mathcal{A}$ has five states belonging 0-circuits and thus, $\ell = 5$. Step (2) then consists in constructing $\mathcal{A}_{(5,?)}$, shown in 5. There is a pseudo-morphism $\mathcal{A} \to \mathcal{A}_{(5,?)}$, whose

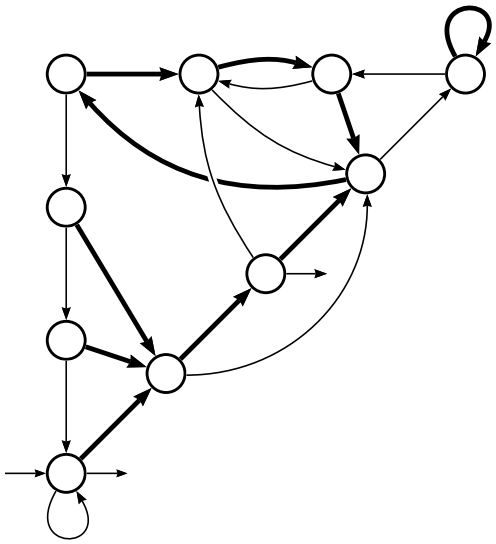


Figure 3 An automaton $\mathcal{A}$.

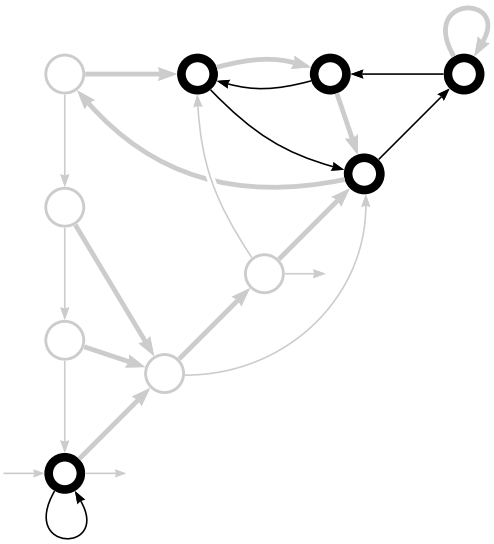


Figure 4 The 0-circuits have 5 states in total.

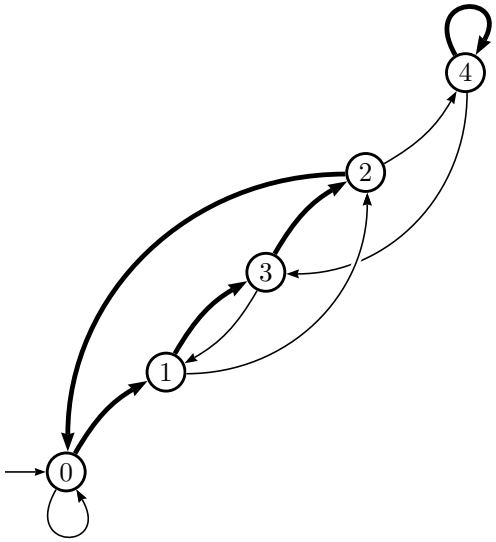


Figure 5 The automaton $\mathcal{A}_{(5,?)}$.

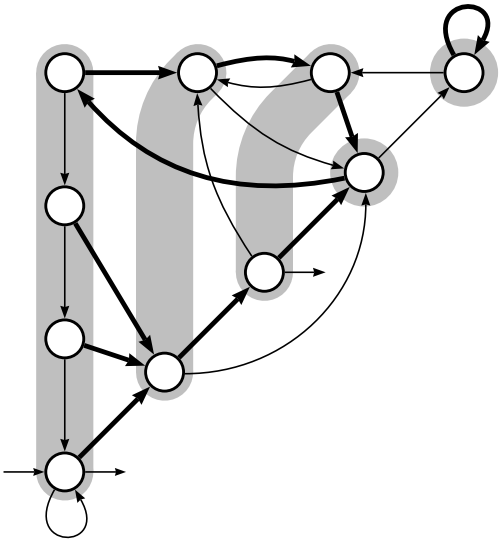


Figure 6 Equivalence classes of the relation induced by the pseudo-morphism $\mathcal{A} \to \mathcal{A}_{(5,?)}$.

equivalence classes are represented in Figure 6. Finally, one could check that Step (4) holds: all states belonging to the same class are 3-ultimately-equivalent. Hence $\mathcal{A}$ accepts an eventually periodic set of period $2^3 \times 5$. It is indeed the minimisation of $\mathcal{A}_{(40,\{0,3\})}$.

5 Generalisation to eventually periodic sets

Let us now consider the eventually periodic sets of integers that are not purely periodic (see Definition 11). We say that such sets are *impurely periodic* and Theorem 28 below gives a characterisation of the minimal automata that accept them.

▶ **Theorem 28.** *Let $b > 1$ be a base and let $\mathcal{A}$ be a minimal automaton over $[\![b]\!]$. We write $(\ell + 1)$ for the number of states in $\mathcal{A}$ that belong to 0-circuits. The automaton $\mathcal{A}$ accepts by value an **im**purely periodic set of integers if and only if the following conditions are met.*

(a) *There exists a pseudo-morphism $\phi : \mathcal{A} \to \mathcal{A}_{(\ell,?)}$.*

(b) *The initial state excluded, the equivalence relation induced by ϕ is a refinement of the ultimate-equivalence relation.*

(c) *The initial state bears a self-loop labelled by the digit 0 and features no other incoming transitions.*

Due to space constraints we do not detail the proof of Theorem 28. Although it is not immediate, it is much similar to the proof of Theorem 23 and may be found in arXiv [8].

As stated by the next corollary, Theorem 28 gives an algorithm to decide whether an automaton accepts an impurely periodic set of integers. It is the same as the one from Section 4.1 with the following modification and addition:

0 to **3.** same tests as in Section 4.1.

4. check whether, for every $x \in \mathbb{Z}/\ell\mathbb{Z}$, the **non-initial** states of $\phi^{-1}(x)$ are ultimately-equivalent;

5. check whether the initial state has no incoming transition.

▶ **Corollary 29.** *Let b be a base and $\mathcal{A}$ be a n-state deterministic automaton over $[\![b]\!]$. It is decidable in $O(bn \log n)$ time whether $\mathcal{A}$ accepts by value an impurely periodic set of integers.*

Since an eventually periodic set is either purely or impurely periodic, Theorem 1 is a direct consequence of Corollaries 25 and 29.

References

1 Jean-Paul Allouche, Narad Rampersad, and Jeffrey Shallit. Periodicity, repetitions, and orbits of an automatic sequence. *Theoret. Comput. Sci*, 410:2795–2803, 2009.

2 Jean-Paul Allouche and Jeffrey Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, 2003.

3 Marie-Pierre Béal and Maxime Crochemore. Minimizing local automata. In M. Fossorier G. Caire, editor, *IEEE Int. Symp. on Information Theory*, pages 1376–1380, 2007.

4 Jason Bell, Emilie Charlier, Aviezri S. Fraenkel, and Michel Rigo. A decision problem for ultimately periodic sets in nonstandard numeration systems. *IJAC*, 19(6):809–839, 2009.

5 Valérie Berthé and Michel Rigo, editors. *Combinatorics, Automata and Number Theory*. Number 135 in Encyclopedia Math. Appl. Cambridge University Press, 2010.

6 Bernard Boigelot and Julien Brusten. A generalization of Cobham's theorem to automata over real numbers. *Theor. Comput. Sci.*, 410(18):1694–1703, 2009.

7 Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Trans. Comput. Log.*, 6(3):614–633, 2005.

8 Bernard Boigelot, Isabelle Mainz, Victor Marsault, and Michel Rigo. An efficient algorithm to decide periodicity of b-recognisable sets using MSDF convention, 2017. Preprint arXiv:1702.03715.

9 V. Bruyère and G. Hansel. Recognizable sets of numbers in nonstandard bases. In R. Baeza-Yates, E. Goles, and P. V. Poblete, editors, *LATIN'95: Theoretical Informatics*, volume 911 of *Lect. Notes Comput. Sci.*, pages 167–179. Springer, 1995.

10 V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc.*, 1:191–238, 1994. Corrigendum, *Bull. Belg. Math. Soc.* **1** (1994), 577.

11 Emilie Charlier, Narad Rampersad, and Jeffrey Shallit. Enumeration and decidable properties of automatic sequences. *Int. J. Found. Comput. Sci.*, 23(5):1035–1066, 2012.

12 Alan Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Mathematical Systems Theory*, 3(2):186–192, 1969. doi:10.1007/BF01746527.

13 Fabien Durand. Decidability of the HD0L ultimate periodicity problem. *RAIRO – Theor. Inf. and Applic.*, 47(2):201–214, 2013.

14 Juha Honkala. A decision method for the recognizability of sets defined by number systems. *ITA*, 20(4):395–403, 1986.

15 Jérôme Leroux. A polynomial time Presburger criterion and synthesis for number decision diagrams. In *LICS 2005*, pages 147–156. IEEE Comp. Soc. Press, 2005.

16 Victor Marsault and Jacques Sakarovitch. Ultimate Periodicity of b-Recognisable Sets: A Quasilinear Procedure. In *DLT 2013*, number 7907 in Lect. Notes Comput. Sci., pages 362–373. Springer, 2013.

17 Ivan Mitrofanov. A proof for the decidability of HD0L ultimate periodicity (in Russian). Preprint arXiv:1110.4780, 2011.

18 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

19 Robert E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

Polynomial-Space Completeness of Reachability for Succinct Branching VASS in Dimension One*

Diego Figueira[1], Ranko Lazić[2], Jérôme Leroux[3], Filip Mazowiecki[4], and Grégoire Sutre[5]

1 **LaBRI, CNRS, Bordeaux, France**
diego.figueira@labri.fr
2 **DIMAP, University of Warwick, Warwick, UK**
R.S.Lazic@warwick.ac.uk
3 **LaBRI, CNRS, Bordeaux, France**
jerome.leroux@labri.fr
4 **DIMAP, University of Warwick, Warwick, UK**
f.mazowiecki@warwick.ac.uk
5 **LaBRI, CNRS, Bordeaux, France**
gregoire.sutre@labri.fr

Abstract

Whether the reachability problem for branching vector addition systems, or equivalently the provability problem for multiplicative exponential linear logic, is decidable has been a long-standing open question. The one-dimensional case is a generalisation of the extensively studied one-counter nets, and it was recently established polynomial-time complete provided counter updates are given in unary. Our main contribution is to determine the complexity when the encoding is binary: polynomial-space complete.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases branching vector addition systems, reachability problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.119

1 Introduction

Background. Vector addition systems, also known as Petri nets (cf., e.g., Reisig's book [21]), are one of the longest established, most extensively studied, and most widely applied models of concurrent computing systems. Their branching generalisation has attracted considerable attention in recent years from the research community on logic in computer science. In addition to the simplicity and elegance of the model, this popularity is due to remarkably close connections with computational linguistics [20, 22], cryptographic protocols [25], linear logic [8, 16], semi-structured databases [13, 1], recursively parallel programs [5], game semantics [7], and timed pushdown systems [6].

A central decision problem for branching vector addition systems is reachability: whether a computation tree exists that has the given root and leaves. Similarly to the simpler setting of Petri nets, this problem has turned out to be very challenging. However, in contrast to Petri nets where the challenge is determining the complexity of reachability below a currently best cubic-Ackermann bound [17], even decidability is still open for the branching vector addition systems reachability problem. For reasons indicated above, the latter question was

* Partially supported by EPSRC grant EP/M011801/1 and Royal Society grant IE150122.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 119; pp. 119:1–119:14

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

Table 1 Complexity of reachability for one-dimensional vector addition systems with states, depending on the presence of branching and the encoding of counter updates.

	unary	binary
1VASS	NL-complete [24, 15]	NP-complete [11]
1BVASS	P-complete [10]	PSPACE-complete

recently highlighted by Bojańczyk as one of a handful of most interesting open problems in computer science logic [4].[1]

The decidability of the branching reachability problem is in fact open already in two dimensions. However, in one dimension, i.e. when there is only one counter, Göller et al. [10] established decidability, and more precisely polynomial-time completeness provided the numbers that specify the counter updates in the system are given in unary. The precise complexity with the encoding in binary remained undetermined.

From another point of view, our investigation builds on the voluminous literature on decision problems for one-counter automata, a ubiquitous class obtained by either dropping one counter from Minsky (two-counter) machines or restricting pushdown automata to one stack symbol. In particular, the complexity of the reachability problem for one-counter systems is known: NL-completeness with the updates given in unary is a classical result [24, 15], and NP-completeness for succinct systems is due to Haase et al. [11] (cf. the latter paper for further references on the subject).

Contributions. Our main result is the closure of the complexity gap for the reachability problem on succinct one-dimensional branching vector addition systems with states (1BVASS), which was between NP hardness inherited from 1VASS [11] and EXPTIME membership that follows from the P membership for unary 1BVASS [10].[2] We show that the problem is in fact PSPACE-complete, which fills the little Table 1.

The fact that the complexities for 1BVASS correspond exactly to 'adding alternation' to the complexities for 1VASS makes them easy to remember. However, it is quite misleading in terms of proofs, at least as far as we can see. The branchings in computations of BVASS are not alternations: counter valuations at child nodes are summed, not compared for equality.[3] Already in the unary case, the proof of P-completeness for 1BVASS [10] is considerably more involved than of NL-completeness for 1VASS [24, 15]. In our proof of PSPACE-completeness for binary 1BVASS, there are several substantial new insights in comparison to both unary 1BVASS and binary 1VASS [11]:

- we introduce a novel notion of implicit reachability witnesses, show that such a witness of at most an exponential size always exists, and hence argue that it can be guessed and checked in polynomial space;
- for the exponential bound on the size of witnesses, a polynomial bound on their counter valuations as for unary 1BVASS [10] is not sufficient because trees with exponentially long branches may be doubly exponentially large;

[1] Although decidability has been stated in a published journal article [2], we believe that claim has not been accepted by the community due to lack of proof, cf. [23, Footnote 4].

[2] We remark that we write 'with states' because stateless (B)VAS are sometimes considered in higher dimensions since states can be encoded at the expense of three additional counters; and that 1VASS, i.e. one-counter nets, are as hard as one-counter systems in this context since the ability to zero-test the counter does not make reachability significantly more complex.

[3] Reachability for alternating VASS is actually undecidable, for relatively trivial reasons [19].

- one of the techniques we employ for establishing the exponential bound involves a novel rewriting strategy, which may be of wider interest since it transforms fragments of computation trees to a normal form that features principal branches, whereas the lack of such a structure has hitherto been an obstacle to generalising Kosaraju's approach [14, 17] to BVASS;
- in contrast to the other three hardness results summarised in Table 1, our lower bound proof is highly intricate, resting on a system of encodings and checks through which alternation can be simulated by additive branching up to a linear depth.

Organisation. After the next section in which we define the systems we consider and observe some of their basic properties, the two sections that follow contain the PSPACE-membership proof. In the penultimate section, we present the PSPACE-hardness construction, and then finish with some concluding remarks.

2 Preliminaries

1BVASS. A *one-dimensional branching vector addition system with states* (*1BVASS* for short) is a triple $B = (Q, \Delta, I)$ where Q is a non-empty finite set of *states*, Δ is a non-empty finite subset of $Q \times Q \times \mathbb{Z} \times Q$, and $I \subseteq Q$ is a finite set of *initial states*. An element $\delta = (q_L, q_R, z, q)$ in Δ is called a *transition*, and the integer z is called the *displacement* of the transition. In the sequel, the maximal absolute displacement is denoted by M. A *configuration* is a pair in $Q \times \mathbb{N}$, and a configuration in $I \times \{0\}$ is called an *initial configuration*. Since the displacements are given in binary, we define the size of B as $|B| = |Q| + |\Delta| \log_2(M+1)$.

Trees. We write $u \preceq v$ if u is a prefix of v and $u \prec v$ if u is a strict prefix. A *tree* is a non-empty finite prefix-closed subset T of $\{L, R\}^*$ satisfying the property that $tL \in T$ if, and only if, $tR \in T$ for every $t \in T$. Elements of T are called *nodes*. Its *root* is the empty word ε. An *ancestor* s of a node t is a prefix of t. In that case t is called a *descendant* of s. By writing *strict* descendants and ancestors we exclude $s = t$. A *child* of a node t is a node tL or tR in T. A node is called a *leaf* if it has no child, and it is said *internal* otherwise. The *sibling* of a node $t \neq \varepsilon$ in the tree T is the node obtained by swapping the last letter. The *height* of a node t is $|t|$. The size of a tree T is its cardinality $|T|$. The *height* of T is the maximal height of any of its nodes. The *subtree* of T rooted at a node t in T is the tree $t^{-1}T = \{t' \in \{L, R\}^* \mid tt' \in T\}$. The *truncation* of T at a node t is the tree $T \backslash t\{L, R\}^+$. Notice that t becomes a leaf of that truncated tree.

Runs and Reachability. We consider labeled trees T where each node t is labeled by a state $q_t \in Q$ and a value $n_t \in \mathbb{N}$ defining a configuration (q_t, n_t). A *run* ρ is a labeled tree such that for every internal node t, there exists an integer z such that (q_{tL}, q_{tR}, z, q_t) is a transition in Δ and such that $n_t = n_{tL} + n_{tR} + z$. The notions of height, size, subtree (called subrun in that context), and truncation are extended from trees to runs as one would expect. Notice that the labels are ignored in the size of a run.

A run is said to be *complete* if every leaf is labeled by an initial configuration. We write *partial run*, instead of run, when we want to emphasize that the run could be not complete. A configuration is said to be *reachable* if it is the root configuration of a complete run. We are mostly interested in the *reachability problem*: given a 1BVASS B and a configuration (q, n) decide whether (q, n) is reachable. The size of the input is $|B| + \log_2(n+1)$.

▶ **Example 1.** Fix numbers n, b such that $0 \leq b \leq 2^n$. We define the 1BVASS $B = (Q, \Delta, I)$, where $Q = \{q_1 \dots q_n\} \cup \{q_I, q_F\}$ and $I = \{q_I\}$. There are three types of transitions:

$$(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1); \quad (q_i, q_i, 0, q_{i+1}) \text{ for all } i < n; \quad (q_n, q_n, -b, q_F).$$

The first two transitions initialize q_1 with 0 or 1; the next n transitions build a full binary tree below each state q_i; and the last transition decreases the value of the counter by b. Consider the reachability problem of $(q_F, 0)$. The complete runs with $(q_F, 0)$ in the root are full binary trees of height $n+1$ such that the number of nodes with state q_1 is 2^n and exactly b of them have value 1.

Contexts and Concatenation. A *context* $\pi = (\rho, t)$ is a run ρ equipped with a distinguished leaf t called the *source* of π. The label of t is called the *source configuration* of π. Such a context is also called a context from the source configuration up to the root configuration of ρ. Given a node t in a run ρ and an ancestor s of t, i.e. such that $t = su$ for some word u, we define the context between (s, t) as the subrun rooted at s of the truncation at t of ρ, equipped with u as the source node. An ancestor of the source t is called a *main node* of π. The set of main nodes of π is called the *main branch*. A *dangling node* in π is a node that is a sibling of a main node. A *dangling configuration* is a configuration of such a node.

The concatenation $\pi\rho$ of a context π with a run ρ is defined if the source configuration (p, m) of π and the root configuration (q, n) of ρ satisfy $p = q$ and if the natural numbers labeling the main nodes of π are larger than or equal to $m - n$. Then $\pi\rho$ is defined by adding to the main nodes of π the integer $n - m$, and replacing the leaf node t of that context with ρ. Notice that $\pi\rho$ is a run. Contexts can be *concatenated* a similar way. The concatenation $\pi\pi'$ of a context $\pi = (\rho, t)$ with a context $\pi' = (\rho', t')$ is defined if $\pi\rho'$ is defined. In that case $\pi\pi'$ is the context $(\pi\rho', tt')$.

Cycles and Minimal Nodes. A context from (p, m) up to (q, n) is called a *cycle* if the source is distinct from the root node and $p = q$. The cycle is said to be simple if on the main branch only the source and the root have the same states. A main node v is said to be *minimal* in a cycle if its value is minimal on the main branch, i.e., $n_v \leq n_{v'}$ for any other main node v'. We write *d-cycle* do emphasize the growth of the cycle, where $d = n - m$. The cycle is said to be *increasing* if $d > 0$, *zero* if $d = 0$, and *decreasing* if $d < 0$.

Let π be a d-cycle, and let p be the state of its source node. Let $\rho = \rho_1\pi\rho_2$ be a context or a run. By *removing* π from ρ we obtain $\rho' = \rho_1\rho_2$ (provided there is no drop below 0). Similarly, let $\rho = \rho_1\rho_2$ be a context or a run such that the source of ρ_1 has state label p. By *inserting* π into v we obtain $\rho_1\pi\rho_2$ (provided that there is no drop below 0). Notice that it is always safe to remove decreasing cycles and to add increasing cycles.

3 d-Coverability

A key role in our polynomial-space algorithm for the reachability problem is played by a more relaxed notion of 'coverability modulo d': instead of reaching a value x exactly, it is allowed to reach any value which is at least x, provided the difference with x is a multiple of d. More formally, a configuration (q, n) is said to be *d-coverable* where $d > 0$ is a natural number if there exists a reachable configuration (q, x) with $x \in n + \mathbb{N}.d$. A *d-coverability run* of a configuration (q, n) is a complete run rooted by a configuration (q, x) with $x \in n + \mathbb{N}.d$. Note that a similar notion of d-reachability was used to show P-completeness for reachability of unary 1BVASS [10].

This section culminates by establishing that every d-coverable configuration (q, n) admits a 'small' d-coverability run, namely of size bounded by $(n+5).(d2^{|B|})^6$. We present most of the proof, which is an orchestration of pigeonhole arguments and safe collapses, as a sequence of lemmas. Let us start with a simple observation about divisibility of subset sums.

▶ **Lemma 2.** *Let $z_1, \ldots, z_d$ be a non-empty sequence of integers. There exists a non-empty finite set $J \subseteq \{1, \ldots, d\}$ such that d divides $\sum_{j \in J} z_j$.*

▶ **Lemma 3.** *Let (q, n) be a configuration and let ρ be a d-coverability run of (q, n) of minimal size. If the size of ρ is larger than $|Q|.2^{|Q|}.d^2$ then ρ contains an increasing cycle.*

Proof. Let ρ be a d-coverability run of (q, n). Suppose: (1) the height of ρ is smaller than $|Q|.d$; and (2) for every height $\ell \geq 1$ the number of nodes in ρ of height ℓ is smaller than $2^{|Q|}.d$. Then it follows that the number of nodes of ρ is bounded by 1 (the root) plus $|Q|.d$ times $2^{|Q|}.d - 1$ (the remaining nodes). Hence the size is bounded by $|Q|.2^{|Q|}.d^2$. It remains to show that if (1) or (2) does not hold then ρ is not minimal or contains an increasing cycle.

In the first case (1) assume that the height of ρ is at least $|Q|.d$. In that case, there exists a node t such that $|t| = |Q|.d$. The nodes on the branch from the root to t are the prefixes of t. It follows that the number of nodes on that branch is equal to $|Q|.d+1$. Notice that if every state of Q occurs at most d times on that branch, then the number of nodes of that branch is bounded by $|Q|.d$ and we get a contradiction. It follows that some state q occurs at least $d+1$ times as a label of a node in that branch. If ρ does not contain any increasing cycle, then all repetitions induce zero or decreasing cycles. Lemma 2 shows that by removing at most d such cycles in that branch we get another d-coverability run of (q, n), smaller than ρ.

In the second case (2) assume that there exists a level $\ell \geq 1$ such that the number of nodes in ρ of height ℓ is at least $2^{|Q|}.d$. It follows that $\ell \geq |Q|$. Notice that these nodes have ancestors in level $\ell - |Q|$. Since the number of elements in level ℓ that have the same ancestors in level $\ell - |Q|$ is bounded by $2^{|Q|}$, it follows that the level $\ell - |Q|$ contains at least d distinct nodes $t_1, \ldots, t_d$ such that, for some words $u_1, \ldots, u_d$ of length $|Q|$, we have $t_i u_i \in \rho$ for every $1 \leq i \leq d$. The pigeon-hole principle shows that we can extract a cycle in the context between $(t_i, t_i u_i)$ for every i. If ρ contains no increasing cycles, then these cycles are zero or decreasing. Lemma 2 shows that by removing at most d such cycles we get another d-coverability run of (q, n), smaller than ρ. ◀

Small d-coverability runs are obtained thanks to the class of *witnesses of d-coverability* defined as follows. A *witness of d-coverability* of a configuration (q, n) is a partial run ψ with the root labeled (q, x), where $x \in n + \mathbb{N}.d$. Every leaf labeled by a configuration (p, m) that is not initial is equipped with a complete run with root label (p, y) with $y \equiv m \mod d$ and containing an increasing cycle. A node of ψ is said to be *modular* when it is an ancestor of such a leaf. We show in the sequel that the existence of d-coverability runs implies the existence of small witnesses of d-coverability. Moreover we provide a way to forge small d-coverability runs from small witnesses of d-coverability.

▶ **Lemma 4.** *Every d-coverable configuration (q, n) has a witness ψ such that:*

- *the subruns of ψ rooted at non-modular nodes have size at most $|Q|.2^{|Q|}.d^2$, and*
- *the complete runs attached to modular leaves have size at most $2|Q|.2^{|Q|}.d^2 + 1$.*

Proof Sketch. We truncate a minimal d-coverability run, bottom-up, at the first increasing cycles. The bounds follow from Lemma 3. ◀

To reduce the number of modular nodes, we introduce an operation on d-coverability witnesses that collapses cycles between modular nodes, as follows. Given a modular leaf ℓ and two ancestors u, v satisfying $u \prec v \preceq \ell$ such that $q_u = q_v$, we transform the witness as follows. First, we introduce the minimal $k \geq 0$ such that $r = kd - n_u + n_v$ is non-negative. Second, we relabel the branch from the root to the leaf ℓ by adding r on nodes s such that $\varepsilon \preceq s \preceq u$ and by adding kd on nodes s such that $v \preceq s \preceq \ell$. It is readily seen that the new labels of u and v are equal. Third, we remove the cycle between u and v by collapsing[4] the nodes $u \prec v$. Notice that after this transformation we get a witness of d-coverability for $(q, n + r)$ where (q, n) was the root label of the original witness of d-coverability. We obtain the following lemma whose proof is along the same lines as that of Lemma 3.

▶ **Lemma 5.** *Let (q, n) be a configuration. By iteratively collapsing cycles, every witness of d-coverability of (q, n) can be simplified into a witness with at most $|Q|.2^{|Q|}.d^2$ modular nodes and where the height of each modular node is smaller than $|Q|.d$.*

▶ **Lemma 6.** *We may relabel modular nodes of any witness of d-coverability of (q, n) in such a way that $n_\ell < n + d + |Q|.d.M$ for every modular leaf ℓ.*

▶ **Theorem 7.** *Every d-coverable configuration (q, n) admits a d-coverability run of size at most $(n + 5).(d2^{|B|})^6$.*

Proof. By applying Lemmas 4, 5, and 6 in succession, we get a witness of d-coverability for (q, n) satisfying the bounds in these lemmas. Assume first that the root node of that witness is not a modular node. In that case the witness of d-coverability of (q, n) is in fact a d-coverability run of (q, n) and by Lemma 4 the size of this run is bounded by $|Q|.2^{|Q|}.d^2$.

Now, suppose that the root node of the witness is a modular node. Let μ be the number of all modular nodes, and ζ the number of all non-modular nodes. By Lemma 5 we get $\mu \leq |Q|.2^{|Q|}.d^2$. We bound ζ as follows. A non-modular node t is called a side node if it is the sibling of a modular node. Observe that non-modular nodes are descendant of side nodes and by Lemma 4 subruns rooted in side nodes have sizes bounded by $|Q|.2^{|Q|}.d^2$. Since the number of side nodes is bounded by μ, we derive that $\zeta \leq \mu.|Q|.2^{|Q|}.d^2$.

To build a d-coverability run from the witness we iterate the following process for each modular leaf ℓ in the witness. As a first step, we transform the attached complete run ρ_ℓ of ℓ in such a way its root label (q_ℓ, x) satisfies $x \in n_\ell + \mathbb{N}d$. Recall that ρ_ℓ contains an increasing cycle π. The above-mentioned transformation simply amounts to iterating this cycle $d.n_\ell$ times. By Lemma 4 the size of ρ_ℓ is bounded by $2.|Q|.2^{|Q|}.d^2 + 1$, which also bounds the size of π. The size of the resulting complete run ρ'_ℓ is bounded by $2.|Q|.2^{|Q|}.d^2 + 1$ (the size of ρ_ℓ) plus $d.n_\ell.2.|Q|.2^{|Q|}.d^2$ (the result of iterating the increasing cycle). It follows that the size of $\rho_{\ell'}$ is bounded by $2.(d.n_\ell + 1).|Q|.2^{|Q|}.d^2 + 1$. By Lemma 6 we have $n_\ell < n + d + |Q|.d.M$, so we get that the size of $\rho_{\ell'}$ is bounded by $2(n + 3).M.|Q|^2.2^{|Q|}.d^4 + 1$.

Let (q_ℓ, m_ℓ) be the configuration of the root of ρ'_ℓ. Observe that $m_\ell \in n_\ell + \mathbb{N}d$. In the second step we add $m_\ell - n_\ell$ to each node on the branch from ℓ to the root of the witness. After this step, the new label of ℓ is equal to the root label of ρ'_ℓ. As a third step, we simply replace the leaf ℓ by the complete run ρ'_ℓ.

We obtain a d-coverability run ρ for (q, n) of size $\mu + \zeta$, plus the sum of the sizes of the complete runs ρ'_ℓ for each modular leaf ℓ. This is bounded by

$$\mu + \mu.|Q|.2^{|Q|}.d^2 + \mu.(2(n + 3).M.|Q|^2.2^{|Q|}.d^4 + 1) \leq (2n + 9).M.|Q|^3.4^{|Q|}.d^6.$$

Since $M.2^{|Q|} \leq 2^{|B|}$, we get that the size of ρ is at most $(n + 5).(d2^{|B|})^6$. ◀

[4] Collapsing two nodes $u \prec v$ consists in replacing the labeled subtree rooted in u by the labeled subtree rooted in v.

4 Reachability

This section is devoted to the proof of the following theorem.

▶ **Theorem 8.** *The reachability problem for 1BVASS is in* PSPACE.

The complexity is w.r.t. the sizes of the input 1BVASS and root configuration, both encoded in binary. Our proof relies on the following small witness property.

▶ **Lemma 9.** *If c is a reachable configuration with a value bounded by $2^{|B|}$ in a given 1BVASS B then there exists a complete run of size at most $2^{60|B|^3}$ with root configuration c.*

Indeed, Lemma 9 implies Theorem 8. First, notice that for configurations with value bigger than $2^{|B|}$ it suffices to solve the problem for value 0 with an auxiliary step in the given BVASS. More precisely, if we ask for reachability of (q, n) we can add two states r, r' such that r is initial, a new transition $(q, r, -n, r')$, and change the question to reachability of $(r', 0)$. To verify if a configuration c with a value bounded by $2^{|B|}$ is reachable, we guess a complete run for c in nondeterministic polynomial space. Since it is impossible to maintain, in polynomial space, all nodes of the run in the memory, we only maintain the ancestors of the currently processed node whose other child was not processed yet. For every processed node v if it is an initial configuration then we go back to the closest ancestor a whose other child was not verified and proceed with that child. In this case we remove the ancestor a from the memory. Otherwise, we guess nondeterministically two children of v such that their triple satisfies some transition in Δ and continue with one of the children. The procedure nondeterministically guesses to proceed with the child whose subtree contains at most half of the leaves in the complete run. It remains to observe that the procedure does not need to remember more than $60|B|^3$ ancestors, otherwise the complete run would require more than $2^{60|B|^3}$ nodes. Notice that it is possible that a node has an exponential number of ancestors, but the procedure does not need to remember them all. The rest of this section is devoted to prove Lemma 9.

Small complete runs are forged from the so-called witnesses of reachability. Formally, the class of *witnesses of reachability* is defined inductively as follows. A witness of reachability w of a configuration c is a partial run with root labeled by c and such that every leaf labeled by (p, m) that is not an initial configuration is a reachable configuration equipped with an *implicit* decreasing simple cycle up to the configuration $(p, 0)$. Implicit means that only the main branch and the dangling nodes of the decreasing cycle are given explicitly. Each dangling configuration is equipped with a witness of reachability. Notice that every configuration admitting a witness is reachable since the leaves of the top most partial run of that witness are labeled by reachable configurations. The *depth* of a witness of reachability is defined as follows. The depth of a complete run is zero, and the depth of a witness of reachability that is not a complete run is one plus the maximal depth of the witnesses of reachability defining the dangling configurations of the decreasing cycles. The depth of a witness of reachability w is denoted by $\mathrm{depth}(w)$. Figure 1 shows an example witness of reachability, suggesting how we turn it into a complete run. To bound the sizes of complete runs obtained from reachability witnesses we introduce the value $\mathrm{maxsize}(w)$ denoting the size of the biggest partial runs occurring in a witness of reachability w. The following lemma shows that $\mathrm{maxsize}(w)$ provides a simple way to bound values occurring in w.

▶ **Lemma 10.** *For every witness of reachability w of a configuration with a value bounded by $2^{|B|}$, the root values of the partial runs used by w are bounded by $2^{|B|}$. Moreover, any value occuring in w is bounded by $2^{2|B|} \cdot \mathrm{maxsize}(w)$.*

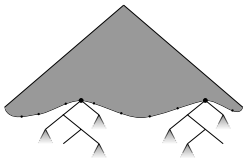


Figure 1 A witness whose topmost partial run has two leaves that are not initial. Decreasing cycles are attached, and the dangling configurations are provided with their subwitnesses.

Proof. The maximal root values can be bounded by observing that, except for the top most partial run of w, partial runs provide root configurations that are dangling configurations of simple decreasing cycles. It follows that these values cannot exceed $|Q|.M \leq 2^{|B|}$. We bound the other values as follows. Observe that the total sum of displacements plus the leave values of a partial run is equal to its root value. It follows that every value of w is bounded by $2^{|B|} + \text{maxsize}(w).M \leq 2^{2|B|}.\,\text{maxsize}(w)$. ◀

▶ **Lemma 11.** *Let w be a witness of reachability of a configuration (q,n) satisfying $n \leq 2^{|B|}$ in a 1BVASS B. There exists a complete run ρ_w with root label (q,n) and size bounded by $(2^{10|B|}.\,\text{maxsize}(w))^{2(\text{depth}(w)+1)}$.*

Proof. We associate to $s, \ell \in \mathbb{N}$ the set $C_{s,\ell}$ of configurations (q,n) such that $n \leq 2^{|B|}$ and such that there exists a witness of reachability w of (q,n) such that $\text{maxsize}(w) \leq s$ and $\text{depth}(w) \leq \ell$. We also define $f(s,\ell) = \max_{c \in C_{s,\ell}}(|\rho_c|)$, where $|\rho_c|$ is the minimal size of a complete run rooted at c. Such a run always exist since c is reachable. Notice that $f(s,0) \leq s$ since a witness of reachability of depth 0 is a complete run.

We provide a bound for $f(s,\ell+1)$ using $f(s,\ell)$. Consider a configuration $c \in C_{s,\ell+1}$. There exists a witness of reachability of c with depth bounded by $\ell+1$ such that $\text{maxsize}(w) \leq s$. Let ρ be the top most partial run of w. Suppose there is a leaf labeled by a non-initial configuration (p,m) that is provided with an implicit simple decreasing cycle up to $(p,0)$. Let us denote by $-d$ the effect of that cycle. As the cycle is simple, it follows that $d \leq |Q|.M \leq 2^{|B|}$. The dangling configurations $c_1, \ldots, c_k$ of that cycle are given by witnesses of reachability $w_1, \ldots, w_k$ such that $\text{depth}(w_j) \leq \ell$ and $\text{maxsize}(w_j) \leq s$. It follows that $c_1, \ldots, c_k \in C_{\ell,s}$. By induction, the dangling configurations $c_1, \ldots, c_k$ can be replaced by complete runs of size bounded by $f(s,\ell)$. Since the cycle is simple, $k \leq |Q|$. After these replacements we obtain an (explicit) simple cycle π of size at most $|Q| + |Q|.f(s,\ell)$. Moreover, since (p,m) is reachable, it is d-coverable. Theorem 7 shows that there exists a d-coverability run for (p,m) of size at most $(m+5).(d.2^{|B|})^6$. Lemma 10 shows that $m \leq 2^{2|B|}.\,\text{maxsize}(w)$. Since $5 \leq 2^{3|B|}.\,\text{maxsize}(w)$ we get $m + 5 \leq 2^{4|B|}.\,\text{maxsize}(w)$. We derive that there is a d-coverability run ρ of (p,m) of size bounded by $\lambda = 2^{16|B|}.\,\text{maxsize}(w) \leq 2^{16|B|}.s$.

There exists $k \in \mathbb{N}$ such that $(p, m + k.d)$ is the root configuration of ρ. In order to obtain a complete run with root configuration (p,m), we just have to consider $\pi^k\rho$. Notice that we can never reach a value below zero since π is a decreasing cycle up to $(p,0)$. As $m + k.d \leq \lambda.M$, it follows that $k \leq \lambda.M$. We have proved that there exists a complete run with root configuration (p,m) and of size bounded by $\lambda.M.(|Q| + |Q|f(s,\ell)) + \lambda \leq 2^{19|B|}.s.f(s,\ell)$. Finally, by replacing every non-terminal leaf by a complete run as performed previously, we get a complete run with root configuration c and size bounded by $2^{19|B|}.s^2.f(s,\ell)$. We have proved that $f(s,\ell+1)$ is bounded by that value. An immediate induction shows that

$$f(s,\ell) \leq (2^{19|B|}.s^2)^\ell.f(s,0) \leq (2^{19|B|}.s^2)^\ell.s \leq (2^{10|B|}.s)^{2(\ell+1)}.$$ ◀

By Lemma 11, to prove Lemma 9 it suffices to find a witness w such that maxsize(w) is bounded exponentially and depth(w) is bounded polynomially in the size of the given 1BVASS. Before we prove that we introduce some notation and two auxiliary lemmas.

Before the next lemma we introduce an operation that intuitively moves increasing cycles from left branches to right branches. By applying that operation as many times as possible, we obtain a so-called *saturated* partial run. Formally, a partial run ρ is said to be *reducible* in a node s if there exists an increasing cycle π between (sL, t) for some node $t \succeq sL$, and a minimal node v in π such that q_v is the state of some descendant of sR. A partial run that is not reducible is said to be *saturated*.

▶ **Lemma 12.** *For every partial run ρ there exists a saturated partial run ρ' with the same root configuration, the same number of nodes, and the same mutiset of leaf configurations.*

Proof. Assume that a partial run ρ is reducible on a node s. Let us denote by π an increasing d-cycle between (sL, t) in ρ for some node $t \succeq sL$ and a minimal node v in π such that $q_v = q_{v'}$ for some v', a descendant of sR. Let $\pi = \pi_1\pi_2$ be such that π_1 is the fragment of π with the source node v. We define $\pi' = \pi_2\pi'_1$, where π'_1 is obtained from π_1 by decreasing all values on the main branch by n_v. Since v is minimal, notice that π' is an increasing d-cycle from $(q_v, 0)$ up to (q_v, d) such that the multiset of configurations of nodes not on the main branch in π and π' are equal. By removing from ρ the increasing cycle π, and inserting the increasing cycle π' into v', we get a partial run ρ' such that the root configuration, the number of nodes, and the mutiset of leaf configurations remain the same as in ρ. Notice that by removing π we decrease the value of s by d, but by inserting π' its value is increased by d, therefore, these operations do not cause a drop below 0. Since this transformation can be performed only a finite number of times, at some point, we get a saturated run satisfying the lemma. ◀

▶ **Lemma 13.** *The number of nodes of a saturated run with root labeled (q, n) with $n \leq 2^{|B|}$ that does not contain any decreasing or zero cycles is bounded by $2^{5|B|^2}$.*

To prove Lemma 9 we will decompose partial runs that contain a decreasing cycle. Since such cycles are not necessarily simple, we provide the following lemma.

▶ **Lemma 14.** *For every decreasing cycle π there exists a state p that labels a main node of π and a simple decreasing cycle π' up to $(p, 0)$ such that the set of dangling configurations of π' is included in the set of dangling configurations of π.*

Proof of Lemma 9. We consider a reachable configuration c with a value bounded by $2^{|B|}$. Obviously c admits a witness of reachability because every complete run is a witness. By Lemma 11 we need to find a witness w of c such that maxsize(w), depth(w) have proper bounds. To do so, we associate to every witness w the sequence of natural numbers $s(w) = (s_j)_{j\geq 1}$, where s_j is the number of partial runs of size j in w, called the *rank* of w. These sequences are ordered colexicographically by the total order $\sqsubseteq$ defined by $(s_j)_{j\geq 1} \sqsubseteq (s'_j)_{j\geq 1}$ if the two sequences are equal or there exists $j \geq 1$ such that $s_j < s'_j$ and $s_i = s'_i$ for every $i > j$. Notice that sequences $s(w)$ have finite support, i.e., $\{j \mid s_j \neq 0\}$ is finite. When restricted to sequences with finite support the order $\sqsubseteq$ is well-founded, i.e., there are no infinite decreasing sequences. We consider for the remainder of the proof a reachability witness w with a minimal rank (for $\sqsubseteq$).

Suppose w has depth $\ell > |Q|$. Then there exists a sequence $\pi_1 \dots \pi_\ell$ of implicit decreasing simple cycles such that π_{i+1} is a cycle equipped to the partial run of a dangling node in π_i. Then there exist $i < j$ such that π_i and π_j have the same state in the root. We replace π_i with

π_j. In particular we remove all cycles $\pi_i \dots \pi_{j-1}$ and all partial runs that were associated to them. The resulting witness has a smaller rank which contradicts our minimality assumption on w.

Now, suppose that w contains partial runs of size bigger than $2^{5|B|^2}$. Let σ be a partial run having the maximal size and such that its depth is maximal (with respect to others of the same size). Notice that all partial runs of larger depth are smaller than σ. Using Lemma 12 we turn σ into a saturated run without changing the multiset of configurations of the leaves. Lemma 10 shows that the run σ has root value at most $2^{|B|}$ and thus by Lemma 13 there exists a decreasing or a zero cycle π in σ. If π is a zero cycle then we just remove it obtaining a smaller rank which contradicts our minimality assumption on w. Otherwise, π is a decreasing cycle. By Lemma 14 there exists a state p that labels a main node u of π and a simple decreasing cycle π' up to $(p, 0)$ such that the set of dangling configurations of π' is included in the set of dangling configurations of π. Since π is a cycle, we can assume w.l.o.g. that u is distinct from the source of π. Let v be the original node of u in σ. By assuption on u, notice that v is an internal node of σ. We define σ' as the partial run truncated at v and equip it with the simple decreasing cycle π'. Since v is an internal node, σ' is smaller from σ. The configurations of dangling nodes in π' are also configurations of dangling nodes in π, which come from σ. We use the partial subruns of σ as partial runs for dangling nodes in π'. These subruns come with implicit simple decreasing cycles and additional partial runs of smaller depth. Notice that, possibly, we have introduced double copies of partial runs of smaller depth. Let us show that the resulting witness w' has a smaller rank which will contradict our minimality assumption on w. Recall that all partial runs of bigger depth are smaller than σ. Since we have decreased the size of σ, and all introduced partial runs are of smaller size than σ it follows that $s(w') \sqsubset s(w)$.

We have proved that $\mathrm{depth}(w) \leq |Q|$ and $\mathrm{maxsize}(w) \leq 2^{5|B|^2}$. From Lemma 11, we get a complete run rooted by (q, n) with size bounded by $2^{60|B|^3}$. ◀

5 Hardness

We prove that the reachability problem for 1BVASS is PSpace-hard. Intuitively, one would like to encode runs of an alternating PTime Turing machine: the tape configuration is maintained as the binary representation of the counter value, and alternation is represented by the branching structure of the run. At first sight, binary rules of BVASS are not compatible with any sort of alternation: the value ℓ of a node may come from two arbitrary values ℓ_1, ℓ_2 from children with the sole restriction that $\ell = \ell_1 + \ell_2$. It thus seems pointless to pretend to replicate information encoded in ℓ into both ℓ_1 and ℓ_2. However, we show that one can enforce —in a highly restricted setting— that transitions behave in a regular way, where $\ell = 2 \cdot \ell_1 = 2 \cdot \ell_2$. Using this, child nodes can recover information from the parent node: the i-th bit of the child has a 1 iff the $i+1$-th bit of the parent has a 1. In this way information can be 'copied' into different branches, and we can benefit from some form of alternation.

▶ **Theorem 15.** *The reachability problem for 1BVASS is* PSpace*-hard.*

Proof Idea. The proof goes by reduction from the PSpace-complete problem of validity for Quantified Boolean Formulas. Given a QBF sentence, such as

$$\varphi = \forall P_1\ \exists P_2\ \forall P_3\ (P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2),$$

we define a polynomial size 1BVASS $\mathcal{B}$, in such a way that the configuration $(q_1, 0)$ is reachable if, and only if, φ is valid. Transitions in $\mathcal{B}$ enforce that any complete run for $(q_1, 0)$ encodes a 'certificate' of the validity of φ. In particular, this certificate contains

- nondeterministic choices for the valuation of existentially quantified variables such as P_2;
- one branch for each of the exponentially-many valuations for universally quantified variables, such as P_1 and P_3;
- for each branch encoding a valuation, a sub-branch for each disjunctive clause, certifying that the clause is true under that valuation.

Levels. For this reduction, it is natural to think of runs as proceeding top-down instead of bottom-up as done hitherto. That is, we start with a configuration $(q_1, 0)$ at the root, and we build valuations going downward until eventually finding an initial configuration on every branch. From this perspective, transitions of $\mathcal{B}$ 'increment' a value $c > 0$ before 'splitting' the value into children with states q', q'' with a transition of the form $(q', q'', -c, q)$. The behaviour of $\mathcal{B}$ ensure that any complete run for $(q_1, 0)$ can be divided into 'levels', so that the i-th level of the tree contains encodings for the choices of valuations for the first i variables of φ. For the purpose of this sketch, the level i is the set of all nodes of height $2 \cdot i$ in the run (*e.g.*, in the run of Figure 2, nodes at level i are those labelled q_{i+1}).

Valuation encoding. Variable valuations are encoded in the counter value by exploiting its compact binary representation, which throughout the run remains always a bitstring of quadratic length in the size of the sentence φ. The counter value bitstring can be split into equal length *segments*, one for each variable, so that the i-th segment is a $2m$ substring encoding the valuation of the i-th variable of φ, where m is the number of universally quantified variables plus the $\log_2$ of the number of conjuncts of φ —in our running example, $m = 3$. The encoding of a valuation for a variable will evolve along the run, for example the encoding for P_1 being true at nodes at different levels may differ. This is because branchings change the counter value and thus its binary representation. For a node at level j, the encoding for a *true* ($\top$) valuation of a variable P_i with $i \leq j$ is through a bitstring $0^{u(j)-1}10^{2m-u(j)}$ at the i-th segment, where $u(j)$ is the number of universally quantified variables P_i with $i \leq j$ in the input sentence —in our example, $u = \{(1,1),(2,1),(3,2)\}$. Similarly, the way to encode a *false* ($\bot$) valuation is through the bitstring $0^{m+u(j)-1}10^{m-u(j)}$. For φ as above, where $m = 3$, the valuation $\{(P_1, \top), (P_2, \bot), (P_3, \bot)\}$ at level 3 (*i.e.*, $j = 3$, $u(j) = 2$) is represented by the bitstring $z = (010|000)(000|010)(000|010)$ (parentheses and pipes are only to improve readability). Let us call the '(i, j)-bit' the j-th most significant bit of the i-th most significant segment in the bitstring, and let $c_{i,j} \in \mathbb{N}$ be the number whose sole (i, j)-bit is 1 in its binary representation (*e.g.*, for z as above, $z = c_{1,2} + c_{2,5} + c_{3,5}$).

As discussed before, for this reduction to work we need that already defined valuation are somehow 'replicated' in all the subtrees, that is, when a configuration branches, information on the valuations is preserved in both children configurations and remains uncorrupted. For this, we enforce that, for every internal node t inside a complete run for $(q_1, 0)$ of $\mathcal{B}$, either:

1. t has a right child with the initial configuration $(q_I, 0)$, or, otherwise,
2. both children of t have the same value, that is, $n_t = 2 \cdot n_{tL} = 2 \cdot n_{tR}$.

Assuming such a property (as verified by the run of Fig. 2), information can be 'spread' along branches of a run: at any node t of type (2) the i-th least significant bit of n_t is 1 iff the $(i-1)$-th least significant bit of n_{tL} and n_{tR} are 1. We use transitions of type (1) to generate a new valuation for the i-th variable, and transitions of type (2) to split the current valuation into two branches. For example, a configuration containing a *true* segment with bitstring $0^{i-1}10^{2m-i}$ is split into two children whose segment value is now 0^i10^{2m-i-1}, which still codes a *true* value for the next level $i + 1$. The choice of m is such that it corresponds

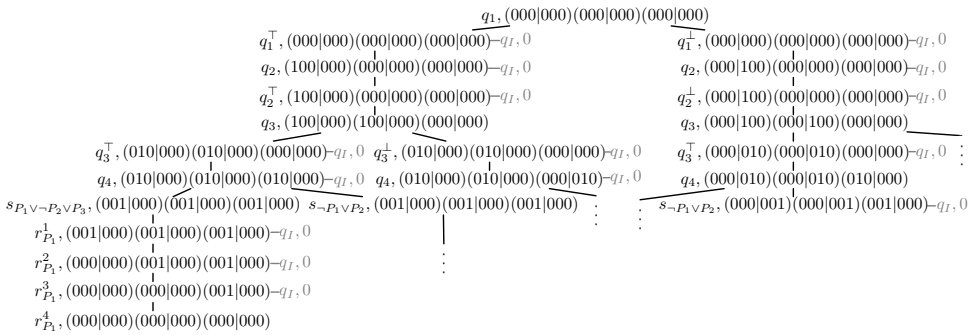


Figure 2 Clipping of a complete run for $(q_1, 0)$. Values are represented by 3 segments of 6 bits.

to the maximum number of transitions of type (2) in any root-to-leaf branch of the run. In other words, m is the maximum distance that a 1-bit can 'travel' along the run.

Here we only show how to build $\mathcal{B}$ for our running example φ. We use the state space $Q = \{q_j, q_i^\top, q_i^\bot, s_\psi, r_A^j, q_I \mid 1 \leq i \leq 3, 1 \leq j \leq 4, \psi : \text{clause}, A : \text{atom}\}$. For each universally quantified variable P_i (*i.e.*, for $i = 1, 3$) we include a transition $(q_i^\top, q_i^\bot, 0, q_i)$, which splits the run into a subtree where P_i is true ($q_i^\top$) and another where it is false ($q_i^\bot$). On the other hand, for each existentially quantified variable P_i (*i.e.*, for $i = 2$), we include non-deterministic transitions $(q_i^\top, q_I, 0, q_i)$ and $(q_i^\bot, q_I, 0, q_i)$, which choose one valuation for P_i. Each state $q_i^\top$ and $q_i^\bot$ has a transition incrementing the corresponding bit in the encoding: $(q_{i+1}, q_I, -c, q_i^\top)$ for c having its $(i, u(i))$-bit in 1, and 0's elsewhere; and $(q_{i+1}, q_I, -\bar{c}, q_i^\bot)$ for $\bar{c}$ having its $(i, m + u(i))$-bit in 1, and 0's elsewhere. Finally, $\mathcal{B}$ checks for the satisfaction of both clauses by splitting the computation through the transition $(s_{P_1 \vee \neg P_2 \vee P_3}, s_{\neg P_1 \vee P_2}, 0, q_4)$. For each clause, $\mathcal{B}$ chooses the atom which will witness its satisfaction, with transitions $(r_A^1, q_I, 0, s_\psi)$ for every disjunctive clause ψ of φ and atom A of ψ (*e.g.*, for $\psi = \neg P_1 \vee P_2$ and $A = \neg P_1$). Finally, the job of r_A^1 is to decrement the (i, m)-bit for verifying that P_i holds true, or the $(i, 2m)$-bit otherwise. However, this choice between the (i, m)- and the $(i, 2m)$-bit must be consistent with the choice of the atom A (*e.g.*, if $A = \neg P_2$ then we must verify that P_2 is false and thus we shouldn't allow the decrement of the (i, m)-bit). Concretely, we include a transition $(r_A^{i+1}, q_I, c_{i,m}, r_A^i)$ if and only if A is not $\neg P_i$; and we include $(r_A^{i+1}, q_I, c_{i,2m}, r_A^i)$ iff A is not P_i. Finally, the initial states I is defined as all states r_A^4 for an atom A, as well as q_I.

Figure 2 contains a depiction of a complete run witnessing the validity of φ. ◀

6 Conclusion

An interesting next question is the complexity of the reachability problem for two-dimensional BVASS, which we conjecture decidable. One approach to establishing decidability could be by generalising the classical algorithm of Hopcroft and Pansiot for two-dimensional VASS [12]. To determine the precise complexity, investigating branching extensions of the flatness notion (cf. [18, 3]) and the cutting technique (cf. [9]) seem like promising directions.

References

1 Sergio Abriola, Diego Figueira, and Santiago Figueira. Logics of repeating values on data trees and branching counter systems. In *FoSSACS*, volume 10203 of *LNCS*. Springer, 2017, to appear.

2 Katalin Bimbó. The decidability of the intensional fragment of classical linear logic. *Theor. Comput. Sci.*, 597:1–17, 2015. doi:10.1016/j.tcs.2015.06.019.

3 Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *LICS*, pages 32–43. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.14.

4 Mikołaj Bojańczyk. Automata column. *SIGLOG News*, 1(2):3–12, 2014. doi:10.1145/2677161.2677163.

5 Ahmed Bouajjani and Michael Emmi. Analysis of recursively parallel programs. *ACM Trans. Program. Lang. Syst.*, 35(3):10:1–10:49, 2013. doi:10.1145/2518188.

6 Lorenzo Clemente, Sławomir Lasota, Ranko Lazić, and Filip Mazowiecki. Timed pushdown automata and branching vector addition systems. Submitted, 2017.

7 Conrad Cotton-Barratt, Andrzej S. Murawski, and C.-H. Luke Ong. ML and extended BVASS. In *ESOP*, 2017. To appear.

8 Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. Vector addition tree automata. In *LICS*, pages 64–73. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319601.

9 Matthias Englert, Ranko Lazić, and Patrick Totzke. Reachability in two-dimensional unary vector addition systems with states is NL-complete. In *LICS*, pages 477–484. ACM, 2016. doi:10.1145/2933575.2933577.

10 Stefan Göller, Christoph Haase, Ranko Lazić, and Patrick Totzke. A polynomial-time algorithm for reachability in branching VASS in dimension one. In *ICALP*, volume 55 of *LIPIcs*, pages 105:1–105:13. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.ICALP.2016.105.

11 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In *CONCUR*, volume 5710 of *LNCS*, pages 369–383. Springer, 2009. doi:10.1007/978-3-642-04081-8_25.

12 John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979. doi:10.1016/0304-3975(79)90041-0.

13 Florent Jacquemard, Luc Segoufin, and Jerémie Dimino. FO2($<$, $+1$, $\sim$) on data trees, data tree automata and branching vector addition systems. *Log. Meth. Comput. Sci.*, 12(2), 2016. doi:10.2168/LMCS-12(2:3)2016.

14 S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *STOC*, pages 267–281. ACM, 1982. doi:10.1145/800070.802201.

15 Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for AC-like equational theories with homomorphisms. Research Report LSV-04-16, ENS de Cachan, 2004.

16 Ranko Lazić and Sylvain Schmitz. Nonelementary complexities for branching VASS, MELL, and extensions. *ACM Trans. Comput. Log.*, 16(3):20:1–20:30, 2015. doi:10.1145/2733375.

17 Jérôme Leroux and Sylvain Schmitz. Ideal decompositions for vector addition systems (invited talk). In *STACS*, volume 47 of *LIPIcs*, pages 1:1–1:13. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.STACS.2016.1.

18 Jérôme Leroux and Grégoire Sutre. On flatness for 2-dimensional vector addition systems with states. In *CONCUR*, volume 3170 of *LNCS*, pages 402–416. Springer, 2004. doi:10.1007/978-3-540-28644-8_26.

19 Patrick Lincoln, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Ann. Pure Appl. Logic*, 56(1-3):239–311, 1992. doi:10.1016/0168-0072(92)90075-B.

20 Owen Rambow. Multiset-valued linear index grammars: Imposing dominance constraints on derivations. In *ACL*, pages 263–270. Morgan Kaufmann Publishers / ACL, 1994. URL: http://aclweb.org/anthology/P/P94/P94-1036.pdf.

21 Wolfgang Reisig. *Understanding Petri Nets – Modeling Techniques, Analysis Methods, Case Studies*. Springer, 2013. doi:10.1007/978-3-642-33278-4.

22 Sylvain Schmitz. On the computational complexity of dominance links in grammatical formalisms. In *ACL*, pages 514–524. The Association for Computer Linguistics, 2010. URL: http://www.aclweb.org/anthology/P10-1053.

23 Sylvain Schmitz. The complexity of reachability in vector addition systems. *SIGLOG News*, 3(1):4–21, 2016. doi:10.1145/2893582.2893585.

24 Leslie G. Valiant and Mike Paterson. Deterministic one-counter automata. *J. Comput. Syst. Sci.*, 10(3):340–350, 1975. doi:10.1016/S0022-0000(75)80005-5.

25 Kumar Neeraj Verma and Jean Goubault-Larrecq. Karp-Miller trees for a branching extension of VASS. *Discr. Math. & Theor. Comput. Sci.*, 7(1):217–230, 2005. URL: http://www.dmtcs.org/volumes/abstracts/dm070113.abs.html.

Satisfiability and Model Checking for the Logic of Sub-Intervals under the Homogeneity Assumption*†

Laura Bozzelli[1], Alberto Molinari[2], Angelo Montanari[3], Adriano Peron[4], and Pietro Sala[5]

1 **University of Napoli "Federico II", Napoli, Italy**
lr.bozzelli@gmail.com
2 **University of Udine, Udine, Italy**
molinari.alberto@gmail.com
3 **University of Udine, Udine, Italy**
angelo.montanari@uniud.it
4 **University of Napoli "Federico II", Napoli, Italy**
adrperon@unina.it
5 **University of Verona, Verona, Italy**
pietro.sala@univr.it

Abstract

In this paper, we investigate the finite satisfiability and model checking problems for the logic D of the sub-interval relation under the homogeneity assumption, that constrains a proposition letter to hold over an interval if and only if it holds over all its points. First, we prove that the satisfiability problem for D, over finite linear orders, is **PSPACE**-complete; then, we show that its model checking problem, over finite Kripke structures, is **PSPACE**-complete as well.

1998 ACM Subject Classification F.4.1 Mathematical Logic, D.2.4 Software/Program Verification

Keywords and phrases Interval Temporal Logic, Satisfiability, Model Checking, Decidability, Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.120

1 Introduction

For a long time, interval temporal logic (ITL) was considered as an attractive, but impractical, alternative to standard point-based ones. On the one hand, as pointed out, among others, by Kamp and Reyle [9], "truth, as it pertains to language in the way we use it, relates sentences not to instants but to temporal intervals", and thus ITL is a natural choice for a specification/representation language; on the other hand, the high undecidability of the satisfiability problem for the most well-known ITLs, such as Halpern and Shoham's HS [7] and Venema's CDT [18], prevented an extensive use of them (in fact, some very restricted variants of them have been successfully applied in formal verification and AI over the years).

* Full version available at http://www.dimi.uniud.it/la-ricerca/pubblicazioni/preprints/4.2017/ [3].

† The work by Alberto Molinari, Angelo Montanari, and Pietro Sala has been supported by the GNCS project *Logic and Automata for Interval Model Checking*.

The recent discovery of a significant number of expressive enough and computationally well-behaved ITLs changed the landscape a lot [6, 13]. Among them, the logic A$\overline{\text{A}}$ of temporal neighborhood [5] and the logic D of (temporal) sub-intervals [4] have a central position. In this paper, we focus on the latter one. D features one modality only, which corresponds to the Allen relation *during* [1]. Since any sub-interval can be defined as an initial sub-interval of an ending one, or, equivalently, as an ending sub-interval of an initial one, it is a (proper) fragment of the logic BE of Allen's relations *started-by* and *finished-by*. From a computational point of view, D is a real character: its satisfiability problem is **PSPACE**-complete over the class of dense linear orders [4, 16] (the problem is undecidable for BE [10]), it becomes undecidable when the logic is interpreted over the classes of finite and discrete linear orders [11], and it is still unknown over the class of all linear orders. As for its expressiveness, unlike A$\overline{\text{A}}$– which is expressively complete with respect to the two-variable fragment of first-order logic for binary relational structures over various linearly-ordered domains [5, 15] – three variables are needed to encode D in first-order logic (the two-variable property is a sufficient condition for decidability, but it is not a necessary one).

In this paper, we show that the decidability of the satisfiability problem for D over the class of finite linear orders can be recovered under the homogeneity assumption (such an assumption constrains a proposition letter to hold over an interval if and only if it holds over all its points). We first prove that the problem belongs to **PSPACE** by exploiting a suitable contraction method. In addition, we prove that the proposed satisfiability checking algorithm can be turned into a **PSPACE** model checking procedure for D formulas over finite Kripke structures (under the homogeneity assumption); **PSPACE**-hardness of both problems follows via a reduction from the language universality problem of nondeterministic finite-state automata. **PSPACE**-completeness of D model checking strongly contrasts with the case of BE, for which only a nonelementary model checking procedure is known [12] and an **EXPSPACE**-hardness result has been given [2].

The rest of the paper is organized as follows. In Section 2, we provide some background knowledge. Then, in Section 3, we prove the **PSPACE** membership of the satisfiability problem for D over finite linear orders (under the homogeneity assumption). Finally, in Section 4, we show that the model checking problem for D over finite Kripke structures (again, under the homogeneity assumption) is in **PSPACE** as well.

All the proofs – here omitted because of lack of space – can be found in [3].

2 The logic D of the sub-interval relation

Let $\mathbb{S} = \langle S, < \rangle$ be a linear order. An *interval* over $\mathbb{S}$ is an ordered pair $[x, y]$, where $x \leq y$. We denote the set of all intervals over $\mathbb{S}$ by $\mathbb{I}(\mathbb{S})$. We consider three possible *sub-interval relations*: (i) the *reflexive* sub-interval relation (denoted as $\sqsubseteq$), defined by $[x, y] \sqsubseteq [x', y']$ iff $x' \leq x$ and $y \leq y'$, (ii) the *proper (or irreflexive)* sub-interval relation (denoted as $\sqsubset$), defined by $[x, y] \sqsubset [x', y']$ iff $[x, y] \sqsubseteq [x', y']$ and $[x, y] \neq [x', y']$, and (iii) the *strict* sub-interval relation (denoted as $\Subset$), defined by $[x, y] \Subset [x', y']$ iff $x' < x$ and $y < y'$.

The three modal logics $\text{D}_{\sqsubseteq}$, $\text{D}_{\sqsubset}$, and $\text{D}_{\Subset}$ feature the same language, consisting of a set $\mathcal{AP}$ of proposition letters/variables, the logical connectives $\neg$ and $\vee$, and the modal operator $\langle D \rangle$. Formally, formulae are defined by the grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle D \rangle \varphi$, with $p \in \mathcal{AP}$. The other connectives, as well as the logical constants $\top$ (true) and $\bot$ (false), are defined as usual; moreover, the dual universal modal operator $[D]\varphi$ is defined as $\neg\langle D \rangle \neg\varphi$. The length of a formula φ, denoted as $|\varphi|$, is the number of sub-formulas of φ.

The semantics of $\text{D}_{\Subset}$, $\text{D}_{\sqsubset}$, and $\text{D}_{\sqsubseteq}$ only differ in the interpretation of the $\langle D \rangle$ operator.

For the sake of brevity, we use $\circ \in \{\sqsubset, \sqsubset, \sqsubseteq\}$ as a shorthand for any of the three sub-interval relations. The semantics of a sub-interval logic $\mathrm{D}_\circ$ is defined in terms of *interval models* $\mathbf{M} = \langle \mathbb{I}(\mathbb{S}), \circ, \mathcal{V} \rangle$. The *valuation function* $\mathcal{V} : \mathcal{AP} \mapsto 2^{\mathbb{I}(\mathbb{S})}$ assigns to every proposition variable p the set of intervals $\mathcal{V}(p)$ over which p holds. The *satisfiability relation* $\models$ is defined as:

- for every proposition letter $p \in \mathcal{AP}$, $\mathbf{M}, [x,y] \models p$ iff $[x,y] \in \mathcal{V}(p)$;
- $\mathbf{M}, [x,y] \models \neg\psi$ iff $\mathbf{M}, [x,y] \not\models \psi$ (i.e., it is not true that $\mathbf{M}, [x,y] \models \psi$);
- $\mathbf{M}, [x,y] \models \psi_1 \vee \psi_2$ iff $\mathbf{M}, [x,y] \models \psi_1$ or $\mathbf{M}, [x,y] \models \psi_2$;
- $\mathbf{M}, [x,y] \models \langle D \rangle \psi$ iff there is an interval $[x',y'] \in \mathbb{I}(\mathbb{S})$ s.t. $[x',y'] \circ [x,y]$ and $\mathbf{M}, [x',y'] \models \psi$.

A $\mathrm{D}_\circ$-formula is $\mathrm{D}_\circ$-*satisfiable* if it holds over some interval of an interval model and it is $\mathrm{D}_\circ$-*valid* if it holds over every interval of every interval model.

In this paper, we restrict our attention to the finite satisfiability problem, that is, satisfiability over the class of finite linear orders. The problem has been shown to be *undecidable* for $\mathrm{D}_\sqsubset$ and $\mathrm{D}_\sqsubset$ [11] and *decidable* for $\mathrm{D}_\sqsubseteq$ [14]. In the following, we show that decidability can be recovered for $\mathrm{D}_\sqsubset$ and $\mathrm{D}_\sqsubset$ by restricting to the class of *homogeneous* interval models. We fully work out the case of $\mathrm{D}_\sqsubset$ (for the sake of simplicity, we will write D for $\mathrm{D}_\sqsubset$), and then we briefly explain how to adapt the proofs to $\mathrm{D}_\sqsubset$.

▶ **Definition 1.** A model $\mathbf{M} = \langle \mathbb{I}(\mathbb{S}), \circ, \mathcal{V} \rangle$ is homogeneous if, for every interval $[x,y] \in \mathbb{I}(\mathbb{S})$ and every $p \in \mathcal{AP}$, it holds that $[x,y] \in \mathcal{V}(p)$ iff $[x',x'] \in \mathcal{V}(p)$ for every $x \leq x' \leq y$.

Hereafter, we will refer to the logic D interpreted over homogeneous models as $\mathrm{D}|_{\mathcal{H}om}$.

2.1 A spatial representation of interval models

We now introduce some basic definitions and notation which will be extensively used in the following. Given a D-formula φ, we define the *closure of* φ, denoted by $\mathrm{CL}(\varphi)$, as the set of all sub-formulas ψ of φ and of their negations $\neg\psi$ (we identify $\neg\neg\psi$ with ψ).

▶ **Definition 2.** Given a D-formula φ, a φ-atom A is a subset of $\mathrm{CL}(\varphi)$ such that: (i) for every $\psi \in \mathrm{CL}(\varphi)$, $\psi \in A$ iff $\neg\psi \notin A$, and (ii) for every $\psi_1 \vee \psi_2 \in \mathrm{CL}(\varphi)$, $\psi_1 \vee \psi_2 \in A$ iff $\psi_1 \in A$ or $\psi_2 \in A$.

The idea underlying atoms is to enforce a "local" (or Boolean) form of consistency among the formulas it contains, that is, a φ-atom A is a *maximal, locally consistent subset* of $\mathrm{CL}(\varphi)$. As an example, $\neg(\psi_1 \vee \psi_2) \in A$ iff $\neg\psi_1 \in A$ and $\neg\psi_2 \in A$. However, note that the definition does not set any constraint on $\langle D \rangle \psi$ formulas, hence the word "local". We denote the set of all φ-atoms as $\mathcal{A}_\varphi$; its cardinality is clearly bounded by $2^{|\varphi|}$ (by point (i) of Definition 2). Atoms are connected by the following binary relation D_φ.

▶ **Definition 3.** Let D_φ be a binary relation over $\mathcal{A}_\varphi$ such that, for each pair of atoms $A, A' \in \mathcal{A}_\varphi$, $A \; D_\varphi \; A'$ holds iff both $\psi \in A'$ and $[D]\psi \in A'$ for each formula $[D]\psi \in A$.

Let A be a φ-atom. We denote by $\mathcal{R}eq_D(A)$ the set $\{\psi \in \mathrm{CL}(\varphi) : \langle D \rangle \psi \in A\}$ of "temporal requests" of A. In particular, if $\psi \notin \mathcal{R}eq_D(A)$, then $[D]\neg\psi \in A$ (by the definition of φ-atom). Moreover, we denote by REQ_φ the set of all arguments of $\langle D \rangle$-formulas in $\mathrm{CL}(\varphi)$, namely, $\mathrm{REQ}_\varphi = \{\psi : \langle D \rangle \psi \in \mathrm{CL}(\varphi)\}$. Finally, we denote by $\mathcal{O}bs_D(A)$ the set $\{\psi \in A : \psi \in \mathrm{REQ}_\varphi\}$ of observables of A. It is easy to prove by induction the next proposition, stating that, once the proposition letters of A and its temporal requests have been fixed, A gets unambiguously determined.

▶ **Proposition 4.** *For any* D*-formula* φ*, given a set* $R \subseteq \mathrm{REQ}_\varphi$ *and a set* $P \subseteq \mathrm{CL}(\varphi) \cap \mathcal{AP}$*, there exists a unique* φ*-atom* A *that satisfies* $\mathcal{R}eq_D(A) = R$ *and* $A \cap \mathcal{AP} = P$*.*

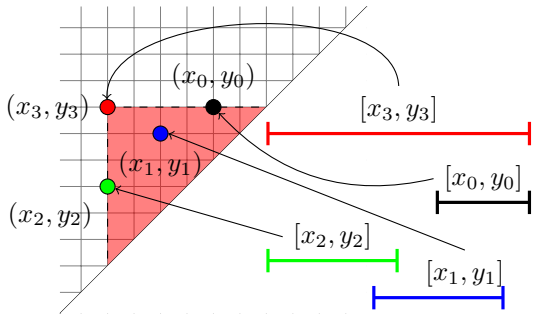


Figure 1 Correspondence between intervals and points of the compass structure.

We now provide a natural interpretation of D over grid-like structures, called *compass structures*, by exploiting the existence of a natural bijection between intervals $[x, y]$ and points (x, y), with $x \leq y$, of an $S \times S$ grid, where $\mathbb{S} = \langle S, < \rangle$ is a finite linear order. Such an interpretation was originally proposed by Venema in [17], and it can also be given for HS and all its (other) fragments.

As an example, Figure 1 shows four intervals $[x_0, y_0], \ldots, [x_3, y_3]$, respectively represented by the points in the grid $(x_0, y_0), \ldots, (x_3, y_3)$, such that: (i) $[x_0, y_0], [x_1, y_1], [x_2, y_2] \sqsubset [x_3, y_3]$, (ii) $[x_1, y_1] \sqsubset\!\!\!\cdot\, [x_3, y_3]$, and (iii) $[x_0, y_0], [x_2, y_2] \not\sqsubset [x_3, y_3]$. The red region highlighted in Figure 1 contains all and only the points (x, y) such that $[x, y] \sqsubset [x_3, y_3]$. Allen interval relation *contains* can thus be represented as a spatial relation between pairs of points. In the following, we make use of $\sqsubset$ also for relating points, i.e., given two points $(x, y), (x', y')$ of the grid, $(x', y') \sqsubset (x, y)$ iff $(x', y') \neq (x, y)$ and $x \leq x' \leq y' \leq y$. Compass structures, repeatedly exploited to establish the following complexity results, can be formally defined as follows.

▶ **Definition 5.** Given a finite linear order $\mathbb{S} = \langle S, < \rangle$ and a D-formula φ, a *compass φ-structure* is a pair $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, where $\mathbb{P}_\mathbb{S}$ is the set of points of the form (x, y), with $x, y \in S$ and $x \leq y$, and $\mathcal{L}$ is a function that maps any point $(x, y) \in \mathbb{P}_\mathbb{S}$ to a φ-atom $\mathcal{L}(x, y)$ in such a way that for all pairs of points $(x, y) \neq (x', y') \in \mathbb{P}_\mathbb{S}$, if $x \leq x' \leq y' \leq y$, then $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x', y')$ (*temporal consistency*).

Due to temporal consistency, the following important property holds in compass structures.

▶ **Lemma 6.** *Given a compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, for all pairs of points $(x', y'), (x, y) \in \mathbb{P}_\mathbb{S}$, if $(x', y') \sqsubset (x, y)$, then $\mathcal{R}eq_D(\mathcal{L}(x', y')) \subseteq \mathcal{R}eq_D(\mathcal{L}(x, y))$ and $\mathcal{O}bs_D(\mathcal{L}(x', y')) \subseteq \mathcal{R}eq_D(\mathcal{L}(x, y))$.*

Fulfilling compass structures are defined as follows.

▶ **Definition 7.** A compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ is said to be *fulfilling* if, for every point $(x, y) \in \mathbb{P}_\mathbb{S}$ and each formula $\psi \in \mathcal{R}eq_D(\mathcal{L}(x, y))$, there exists a point $(x', y') \sqsubset (x, y)$ in $\mathbb{P}_\mathbb{S}$ such that $\psi \in \mathcal{L}(x', y')$.

Note that if $\mathcal{G}$ is fulfilling, then $\mathcal{R}eq_D(\mathcal{L}(x, x)) = \emptyset$ for all points "on the diagonal" $(x, x) \in \mathbb{P}_\mathbb{S}$.

We say that a compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ *features* a formula ψ if there exists a point $(x, y) \in \mathbb{P}_\mathbb{S}$ such that $\psi \in \mathcal{L}(x, y)$. The following result holds.

▶ **Proposition 8.** *A* D*-formula φ is satisfiable iff there is a fulfilling compass φ-structure that features it.*

In a fulfilling compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, where $S = \{0, \ldots, t\}$, w.l.o.g., we will sometimes assume φ to be satisfied by the maximal interval $[0, t]$, that is, $\varphi \in \mathcal{L}(0, t)$.

The notion of homogeneous models directly transfers to compass structures.

▶ **Definition 9.** A compass φ-structure $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$ is *homogeneous* if, for every point $(x, y) \in \mathbb{P}_{\mathbb{S}}$ and each $p \in \mathcal{AP}$, we have that $p \in \mathcal{L}(x, y)$ iff $p \in \mathcal{L}(x', x')$ for all $x \leq x' \leq y$.

Proposition 8 can be tailored to homogeneous compass structures as follows.

▶ **Proposition 10.** *A* $\mathrm{D}|_{\mathcal{H}om}$*-formula* φ *is satisfiable iff there is a fulfilling homogeneous compass* φ*-structure that features it.*

3 Satisfiability of $\mathrm{D}|_{\mathcal{H}om}$ over finite linear orders

In this section, we devise a satisfiability checking procedure for $\mathrm{D}|_{\mathcal{H}om}$-formulas over finite linear orders, which will also allow us to easily derive a model checking algorithm for $\mathrm{D}|_{\mathcal{H}om}$ over finite Kripke structures. To start with, we show that there is a ternary relation between φ-atoms, that we denote by $\mathrel{=}^{D_\varphi}\!\!\rightarrow$, such that if it holds among all atoms in consecutive positions of a compass φ-structure, then the structure is fulfilling. Hence, we may say that $\mathrel{=}^{D_\varphi}\!\!\rightarrow$ is the rule for labeling fulfilling compasses. Next, we introduce an equivalence relation $\sim$ between *rows* of a compass φ-structure. Since it has finite index – exponentially bounded by $|\varphi|$ – and it preserves fulfillment of compasses, it is intuitively possible to "contract" the structures when we can find two related rows. Moreover, any contraction done according to $\sim$ keeps the same atoms (only the number of their occurrences may vary), and thus if a compass features φ before the contraction, then φ is still featured after it. This fact is exploited to build a satisfiability algorithm for $\mathrm{D}|_{\mathcal{H}om}$-formulas which makes use of *polynomial working space* only, because (i) it only needs to keep track of two rows of a compass at a time, (ii) all rows satisfy some nice properties that make it possible to succinctly encode them, and (iii) compass contractions are implicitly performed by means of a reachability check in a suitable graph, whose nodes are the equivalence classes of $\sim$.

Let us now introduce the aforementioned ternary relation $\mathrel{=}^{D_\varphi}\!\!\rightarrow$ among atoms.

▶ **Definition 11.** Given three φ-atoms A_1, A_2 and A_3, we say that A_3 is D_φ-generated by A_1, A_2 (written $A_1 A_2 \mathrel{=}^{D_\varphi}\!\!\rightarrow A_3$) if: (i) $A_3 \cap \mathcal{AP} = A_1 \cap A_2 \cap \mathcal{AP}$ and (ii) $\mathcal{R}eq_D(A_3) = \mathcal{R}eq_D(A_1) \cup \mathcal{R}eq_D(A_2) \cup \mathcal{O}bs_D(A_1) \cup \mathcal{O}bs_D(A_2)$.

It is immediate to check that $A_1 A_2 \mathrel{=}^{D_\varphi}\!\!\rightarrow A_3$ iff $A_2 A_1 \mathrel{=}^{D_\varphi}\!\!\rightarrow A_3$, that is, the order of the first two components in the ternary relation is irrelevant. The next result, following from Proposition 4, proves that $\mathrel{=}^{D_\varphi}\!\!\rightarrow$ expresses a *functional dependency* on φ-atoms.

▶ **Lemma 12.** *Given two* φ*-atoms* $A_1, A_2 \in \mathcal{A}_\varphi$*, there exists* exactly one φ*-atom* $A_3 \in \mathcal{A}_\varphi$ *such that* $A_1 A_2 \mathrel{=}^{D_\varphi}\!\!\rightarrow A_3$*.*

Definition 11 and Lemma 12 can be exploited to label a homogeneous compass φ-structure $\mathcal{G}$, namely, to determine the φ-atoms labeling all the points (x, y) of $\mathcal{G}$, starting from the ones on the diagonal. The idea is the following: if two φ-atoms A_1 and A_2 label respectively the greatest proper prefix $[x, y-1]$, that is, the point $(x, y-1)$, and the greatest proper suffix $[x+1, y]$, that is, $(x+1, y)$, of the same interval $[x, y]$, then the atom A_3 labeling $[x, y]$ is unique, and it is precisely the one satisfying $A_1 A_2 \mathrel{=}^{D_\varphi}\!\!\rightarrow A_3$ (see Figure 2). The next lemma proves that this is the general rule for labeling fulfilling homogeneous compasses.

▶ **Lemma 13.** *Let* $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$*.* $\mathcal{G}$ *is a fulfilling homogeneous compass* φ*-structure iff, for every pair* $x, y \in S$*, we have: (i)* $\mathcal{L}(x, y-1)\mathcal{L}(x+1, y) \mathrel{=}^{D_\varphi}\!\!\rightarrow \mathcal{L}(x, y)$ *if* $x < y$*, and (ii)* $\mathcal{R}eq_D(\mathcal{L}(x, y)) = \emptyset$ *if* $x = y$*.*

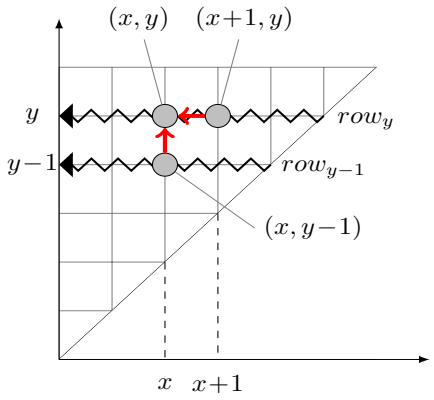


Figure 2 Rule for labeling homogeneous fulfilling compass φ-structures.

Now we introduce the concept of φ-row, which can be viewed as the ordered sequence of (the occurrences of) atoms labelling a row of a compass φ-structure. Given an atom $A \in \mathcal{A}_\varphi$, we call it *reflexive* if $A\ D_\varphi\ A$, *irreflexive* otherwise.

▶ **Definition 14.** A φ-row is a finite sequence of φ-atoms $row = A_0^{m_0} \cdots A_n^{m_n}$, where A^m stands for m repetitions of A, such that for each $0 \leq i \leq n$, we have that $m_i > 0$ – if $m_i > 1$, then A_i is reflexive – and for each $0 \leq j < i$, it holds that $A_i\ D_\varphi\ A_j$, $A_i \neq A_j$, and $(A_j \cap \mathcal{AP}) \supseteq (A_i \cap \mathcal{AP})$. Moreover, $\mathcal{R}eq_D(A_0) = \emptyset$.

The length of a φ-row $row = A_0^{m_0} \cdots A_n^{m_n}$ is defined as $|row| = \sum_{0 \leq i \leq n} m_i$, and for each $0 \leq j < |row|$, the j-th element, denoted by $row[j]$, is the j-th symbol in the word $A_0^{m_0} \cdots A_n^{m_n}$, e.g., $row[0] = A_0$, $row[m_0] = A_1$, ... We denote by $\mathcal{R}ows_\varphi$ the set of all possible φ-rows. This set may be infinite.

The number of distinct atoms in any φ-row is bounded. Since for each $0 \leq i \leq n$ and each $0 \leq j < i$, $A_i\ D_\varphi\ A_j$, it holds that $\mathcal{R}eq_D(A_j) \subseteq \mathcal{R}eq_D(A_i)$. Therefore, two monotonic sequences for every φ-row can be considered, one increasing, i.e., $\emptyset = \mathcal{R}eq_D(A_0) \subseteq \mathcal{R}eq_D(A_1) \subseteq \ldots \subseteq \mathcal{R}eq_D(A_n)$, and one decreasing, i.e., $(A_0 \cap \mathcal{AP}) \supseteq (A_1 \cap \mathcal{AP}) \supseteq \ldots \supseteq (A_n \cap \mathcal{AP})$. The number of distinct elements is bounded by $|\varphi|$ in the former sequence and by $|\varphi| + 1$ in the latter (as $|\,\mathrm{REQ}_\varphi\,| \leq |\varphi| - 1$ and $|\mathcal{AP}| \leq |\varphi|$–w.l.o.g., we can consider only the letters actually occurring in φ). Since, as already shown (Proposition 4), a set of requests and a set of proposition letters uniquely determine a φ-atom, any φ-row may feature at most $2|\varphi|$ distinct atoms, i.e., $n < 2|\varphi|$.

Given a homogeneous compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, for every $y \in S$, we define row_y as the word of φ-atoms $row_y = \mathcal{L}(y, y) \cdots \mathcal{L}(0, y)$, i.e., the sequence of atoms labeling points of $\mathcal{G}$ with the same y-coordinate, starting from the one on the diagonal inwards (see Figure 2).

▶ **Lemma 15.** *Let $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ be a fulfilling homogeneous compass φ-structure. For every $y \in S$, row_y is a φ-row.*

We now define the *successor* relation between pairs of φ-rows, denoted as $=^{row_\varphi}\!\!\rightarrow$, which is basically a component-wise application of $=^{D_\varphi}\!\!\rightarrow$ over the elements of two φ-rows (remember that atoms on rows are collected from right to left).

▶ **Definition 16.** Given two φ-rows row and row', we say that row' is a *successor* of row, or $row =^{row_\varphi}\!\!\rightarrow row'$, if $|row'| = |row| + 1$, and for all $0 \leq i < |row|$, $row[i] row'[i] =^{D_\varphi}\!\!\rightarrow row'[i+1]$.

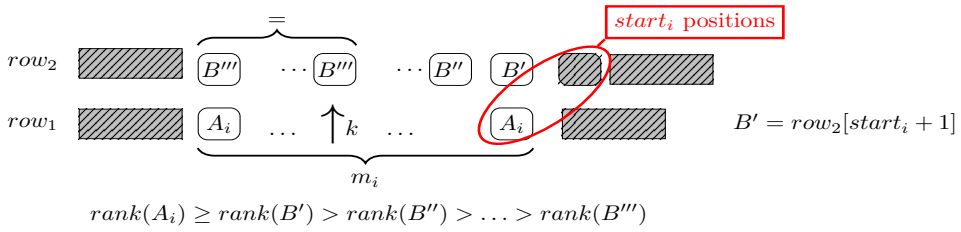


Figure 3 A graphical account of the proof of Lemma 18.

The next lemma states that consecutive rows in homogeneous fulfilling compass φ-structures respect the successor relation.

▶ **Lemma 17.** *Let $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, with $\mathcal{R}eq_D(\mathcal{L}(x,x)) = \emptyset$ for all $(x,x) \in \mathbb{P}_\mathbb{S}$. $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure iff, for each $0 \leq y < |S| - 1$, $row_y =^{row_\varphi}\!\!\Rightarrow row_{y+1}$.*

Given an atom $A \in \mathcal{A}_\varphi$, we define the *rank of A*, written $rank(A)$, as $|\,\mathrm{REQ}_\varphi\,| - |\mathcal{R}eq_D(A)|$. Clearly, $rank(A) < |\varphi|$. Whenever $A\ D_\varphi\ A'$, for some $A' \in \mathcal{A}_\varphi$, $\mathcal{R}eq_D(A') \subseteq \mathcal{R}eq_D(A)$, and hence $rank(A) \leq rank(A')$ and $|\mathcal{R}eq_D(A) \setminus \mathcal{R}eq_D(A')| \leq rank(A')$. We can see the *rank* of an atom as the "number of degrees of freedom" that it gives to the atoms that stay "above it". In particular, by definition, for every φ-row $row = A_0^{m_0} \cdots A_n^{m_n}$, we have $rank(A_0) \geq \ldots \geq rank(A_n)$. The next result uses the notion of rank to provide an insight on how consecutive φ-rows are connected (see Figure 3).

▶ **Lemma 18.** *Let row_1, row_2 be two φ-rows, with $row_1 = A_0^{m_0} \cdots A_n^{m_n}$ and $row_1 =^{row_\varphi}\!\!\Rightarrow row_2$. For each $0 \leq i \leq n$, let $start_i = \sum_{0 \leq j < i} m_j$. If $m_i > rank(A_i)$, then there exists $start_i < k \leq start_i + m_i$ such that: (i) $row_2[k]$ is reflexive; (ii) $rank(row_2[j]) > rank(row_2[j+1])$ for each $start_i < j < k$; (iii) $row_2[j] = row_2[j+1]$ for each $k \leq j < start_i + m_i$; (iv) if m' is the exponent of the atom $row_2[k]$, then $m' > rank(row_2[k])$.*

Proof. If $m_i = 1$, by hypothesis we have $rank(A_i) = 0$. Hence, $rank(row_2[start_i + 1]) = 0$, because $row_1 =^{row_\varphi}\!\!\Rightarrow row_2$, and thus $row_2[start_i + 1]$ is (trivially) reflexive. All claims hold by choosing $k = start_i + 1$.

Let us then assume $m_i > 1$. First, we prove that for each $start_i < j \leq start_i + m_i$, if $row_2[j]$ is reflexive, then for each $j \leq j' \leq start_i + m_i$, $row_2[j'] = row_2[j]$. If $j = start_i + m_i$ there is nothing to prove. Thus, let us consider $j < start_i + m_i$. Since we are assuming that $row_2[j]$ is reflexive, then $\mathcal{O}bs_D(row_2[j]) \subseteq \mathcal{R}eq_D(row_2[j])$. Since $row_1 =^{row_\varphi}\!\!\Rightarrow row_2$, we have that $\mathcal{R}eq_D(A_i), \mathcal{O}bs_D(A_i) \subseteq \mathcal{R}eq_D(row_2[j])$, and $\mathcal{R}eq_D(row_2[j+1]) = \mathcal{R}eq_D(row_2[j]) \cup \mathcal{O}bs_D(row_2[j]) \cup \mathcal{R}eq_D(A_i) \cup \mathcal{O}bs_D(A_i) = \mathcal{R}eq_D(row_2[j])$. Moreover, again from $row_1 =^{row_\varphi}\!\!\Rightarrow row_2$, we have that $row_2[j] \cap \mathcal{AP} = row_2[j-1] \cap A_i \cap \mathcal{AP}$ and $row_2[j+1] \cap \mathcal{AP} = row_2[j] \cap A_i \cap \mathcal{AP} = row_2[j-1] \cap A_i \cap \mathcal{AP}$. Thus, $row_2[j+1] = row_2[j]$, because the two atoms feature exactly the same requests and proposition letters (Proposition 4). Then, since $A_i\ row_2[j] =^{D_\varphi}\!\!\Rightarrow row_2[j+1]$, by iterating the reasoning and exploiting Lemma 12 we can conclude that $row_2[j] = row_2[j']$ for each $j \leq j' \leq start_i + m_i$.

Now, it can be easily shown that if we have two atoms A and A' such that $A\ D_\varphi\ A'$ and A' is irreflexive, then $rank(A) < rank(A')$, and we have just proved that we cannot interleave reflexive atoms with irreflexive ones "above" the A_i's (all irreflexive atoms must "come before" reflexive ones in the part of row_2 "above" the A_i's). Thus, in the worst possible case, the atoms $row_2[start_i + 1], \ldots, row_2[start_i + rank(A_i)]$ may be irreflexive (as $rank(row_2[start_i + 1]) > \ldots > rank(row_2[start_i + rank(A_i)])$ and $rank(A_i) \geq rank(row_2[start_i + 1])$). Note that these irreflexive atoms may be the "first" $rank(A_i)$ atoms above the A_i's only, and not the

"first" $rank(A_i)+1$, since any atom with rank equal to 0 is reflexive. We conclude that $row_2[start_i+rank(A_i)+1]$ must be reflexive. Thus, we can choose $k = start_i+rank(A_i)+1$. Since by hypothesis $m_i \geq rank(A_i)+1$, we get that $start_i < k \leq start_i + m_i$.

As for the last claim, we have that $rank(row_2[k]) \leq rank(row_2[start_i+1]) - (k - start_i - 1) \leq rank(A_i) - (k - start_i - 1)$. Then, the exponent m' of $row_2[k]$ is such that $m' \geq m_i - (rank(A_i) - rank(row_2[k]))$, that is, at least $m_i - (rank(A_i) - rank(row_2[k]))$ atoms labelled by $row_2[k]$ occur in the block $start_i+1, \ldots, start_i+m_i$ of row_2 (see Figure 3). Since by hypothesis $m_i > rank(A_i)$, then $m_i - rank(A_i) > 0$ and $rank(row_2[k]) < m'$. ◀

Now we introduce an equivalence relation $\sim$ over $\mathcal{R}ows_\varphi$ which is the key ingredient of the proofs showing that both satisfiability and MC for D$|_{\mathcal{H}om}$-formulas are decidable.

▶ **Definition 19.** Given two φ-rows $row_1 = A_0^{m_0} \cdots A_n^{m_n}$ and $row_2 = \hat{A}_0^{\hat{m}_0} \cdots \hat{A}_{\hat{n}}^{\hat{m}_{\hat{n}}}$, we say that they are *equivalent*, written $row_1 \sim row_2$, if (i) $n = \hat{n}$, and (ii) for each $0 \leq i \leq n$, $A_i = \hat{A}_i$, and $m_i = \hat{m}_i$ or both m_i and $\hat{m}_i$ are (strictly) greater than $rank(A_i)$.

Note that if two rows feature the same set of atoms, the lower the rank of an atom A_i, the lower the number of occurrences of A_i both the rows have to feature in order to belong to the same equivalence class. As an example, let row_1 and row_2 be two rows with $row_1 = A_0^{m_0}A_1^{m_1}$, $row_2 = A_0^{\overline{m}_0}A_1^{\overline{m}_1}$, $rank(A_0) = 4$, and $rank(A_1) = 3$. If $m_1 = 4$ and $\overline{m}_1 = 5$ they are both greater than $rank(A_1)$, and hence they do not violate the condition for $row_1 \sim row_2$. On the other hand, if $m_0 = 4$ and $\overline{m}_0 = 5$, we have that m_0 is less than or equal to $rank(A_0)$. Thus, in this case, $row_1 \not\sim row_2$ due to the indexes of A_0. This happens because $rank(A_0)$ is greater than $rank(A_1)$. Two cases in which $row_1 \sim row_2$ are $m_0 = \overline{m}_0$ and $m_0, \overline{m}_0 \geq 5$.

The relation $\sim$ has finite index, which is roughly bounded by the number of all the possible φ-rows $row = A_0^{m_0} \cdots A_n^{m_n}$, with exponents m_i ranging from 1 to $|\varphi|$. Since (i) the number of possible atoms is $2^{|\varphi|}$, (ii) the number of *distinct* atoms in any φ-row is at most $2|\varphi|$, and (iii) the number of possible functions $f : \{1, \ldots, \ell\} \to \{1, \ldots, |\varphi|\}$ is $|\varphi|^\ell$, we have that the number of distinct equivalence classes of $\sim$ is bounded by

$$\sum_{j=1}^{2|\varphi|} (2^{|\varphi|})^j \cdot |\varphi|^j \leq 2^{3|\varphi|^2},$$

which is exponential in the length of the input formula φ. We denote the set of the equivalence classes of $\sim$ over all the possible φ-rows by $\mathcal{R}ows_\varphi^\sim$.

Now we extend the relation $=^{row_\varphi}\!\!\Rightarrow$ to equivalence classes of $\sim$ in the following way.

▶ **Definition 20.** Given two φ-row classes $[row_1]_\sim$ and $[row_2]_\sim$, we say that $[row_2]_\sim$ is a successor of $[row_1]_\sim$, written $[row_1]_\sim =^{row_\varphi}\!\!\Rightarrow [row_2]_\sim$, if there exist $row_1' \in [row_1]_\sim$ and $row_2' \in [row_2]_\sim$ such that $row_1' =^{row_\varphi}\!\!\Rightarrow row_2'$.

The following result proves that if some $row_1' \in [row_1]_\sim$ has a successor in $[row_2]_\sim$, then *every* φ*-row* of $[row_1]_\sim$ has a successor in $[row_2]_\sim$.

▶ **Lemma 21.** *Given two* φ*-row classes* $[row_1]_\sim$ *and* $[row_2]_\sim$ *such that* $[row_1]_\sim =^{row_\varphi}\!\!\Rightarrow [row_2]_\sim$, for every $row \in [row_1]_\sim$ *there exists* $row' \in [row_2]_\sim$ *such that* $row =^{row_\varphi}\!\!\Rightarrow row'$.

The proof, omitted for space reasons, begins by considering two φ-rows, row and $\overline{row}$, such that $row \in [row_1]_\sim$, $\overline{row} \in [row_2]_\sim$, and $row =^{row_\varphi}\!\!\Rightarrow \overline{row}$ (such a pair always exists by Definition 20). Then, we consider another φ-row, $row' \neq row$ in $[row_1]_\sim$, and we show (constructively) how to build $\overline{row}' \in [row_2]_\sim$ such that $row' =^{row_\varphi}\!\!\Rightarrow \overline{row}'$. This is sufficient to

Input: a D$|_{\mathcal{H}om}$-formula φ

1. Put $M \leftarrow 2^{3|\varphi|^2}$, $step \leftarrow 0$ and $row \leftarrow A$ for some atom $A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$.
2. If there exists $0 \leq i < |row|$ such that $\varphi \in row[i]$, return **satisfiable**.
3. If $step = M - 1$, return **unsatisfiable**.
4. Non-deterministically generate a φ-row row' and check that $row =^{row_\varphi}\!\!\Rightarrow row'$.
5. Put $step \leftarrow step + 1$ and $row \leftarrow row'$.
6. Go back to 2.

Figure 4 Non-deterministic procedure deciding the satisfiability of a D$|_{\mathcal{H}om}$-formula φ.

prove the claim: $\overline{row}'$ is built by making use of the facts that $row' \sim row$ and $row =^{row_\varphi}\!\!\Rightarrow \overline{row}$, and of the properties stated by Lemma 18.

The following result arranges the equivalence classes $\mathcal{R}ows_\varphi^\sim$ in a graph $G_{\varphi\sim}$.

▶ **Definition 22.** Let φ be a D$|_{\mathcal{H}om}$-formula. The $\varphi\sim$*graph* of φ is the graph $G_{\varphi\sim} = (\mathcal{R}ows_\varphi^\sim, =^{row_\varphi}\!\!\Rightarrow)$.

The next theorem reduces the problem of satisfiability checking for a D$|_{\mathcal{H}om}$-formula φ over finite linear orders (equivalent, by Proposition 10, to deciding if there is a homogeneous fulfilling compass φ-structure that features φ) to a reachability problem in the $\varphi\sim$graph, allowing us to determine the computational complexity of the former problem.

▶ **Theorem 23.** *Given a* D$|_{\mathcal{H}om}$*-formula φ, there exists a homogeneous fulfilling compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ that features φ iff there exists a path in $G_{\varphi\sim} = (\mathcal{R}ows_\varphi^\sim, =^{row_\varphi}\!\!\Rightarrow)$ from some class $[row]_\sim \in \mathcal{R}ows_\varphi^\sim$ to some class $[row']_\sim \in \mathcal{R}ows_\varphi^\sim$ such that* (1) *there exists $row_1 \in [row]_\sim$ with $|row_1| = 1$, and* (2) *there exist $row_2 \in [row']_\sim$ and $0 \leq i < |row_2|$ such that $\varphi \in row_2[i]$.*

Proof. Preliminarily we observe that, in (1), if $|row_1| = 1$, then $\{row_1\} = [row]_\sim$; moreover, in (2), if for $row_2 \in [row']_\sim$ and $0 \leq i < |row_2|$ we have that $\varphi \in row_2[i]$, then for any $row_2' \in [row']_\sim$, there is $0 \leq i' < |row_2'|$ such that $\varphi \in row_2'[i']$.

($\Rightarrow$) Let us consider a homogeneous fulfilling compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ that features φ. By Lemmata 15 and 17, $\mathcal{L}(0,0) =^{row_\varphi}\!\!\Rightarrow row_1 =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow row_{\max(S)}$. Thus there exist two indexes $0 \leq j \leq \max(S)$ and $0 \leq i < |row_j|$ for which $\varphi \in row_j[i]$. By Definition 20, we get that $[\mathcal{L}(0,0)]_\sim =^{row_\varphi}\!\!\Rightarrow [row_1]_\sim =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow [row_j]_\sim$ is a path in $G_{\varphi\sim}$; it is immediate to check that it fulfils requirements (1) and (2).

($\Leftarrow$) Let us assume there exists a path $[row_0]_\sim =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow [row_m]_\sim$ in $G_{\varphi\sim} = (\mathcal{R}ows_\varphi^\sim, =^{row_\varphi}\!\!\Rightarrow)$ for which $|row_0| = 1$ and there exists i such that $\varphi \in row_m[i]$. By applying repeatedly Lemma 21 we get that there exists a sequence $row_0' =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow row_m'$ of φ-rows where $row_0' = row_0$, for every $0 \leq j \leq m$, $row_j' \in [row_j]_\sim$, and there exists i' such that $\varphi \in row_m'[i']$. We observe that, by Definition 16, $|row_j'| = |row_{j-1}'| + 1$ for $1 \leq j \leq m$ and, since $|row_0'| = 1$, we have $|row_j'| = j + 1$. Let us now define $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ where $S = \{0, \ldots, m\}$ and $\mathcal{L}(x,y) = row_y'[y - x]$ for every $0 \leq x \leq y \leq m$. By Lemma 17, $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure. Finally, since $\varphi \in row_m'[i']$, $\mathcal{G}$ features φ. ◀

The size of $G_{\varphi\sim} = (\mathcal{R}ows_\varphi^\sim, =^{row_\varphi}\!\!\Rightarrow)$ is bounded by $|\mathcal{R}ows_\varphi^\sim|^2$, which is exponential in $|\varphi|$. However, it is possible to (non-deterministically) perform a reachability in $G_{\varphi\sim}$ by using space *logarithmic* in $|\mathcal{R}ows_\varphi^\sim|^2$. The *non-deterministic* procedure of Figure 4 exploits this fact in order to decide the satisfiability of a D$|_{\mathcal{H}om}$-formula φ, by using only a working space

polynomial in $|\varphi|$: it searches for a suitable path in $G_{\varphi\sim}$, $[row_0]_\sim \Rightarrow^{row_\varphi} \cdots \Rightarrow^{row_\varphi} [row_m]_\sim$, where $row_0 = A$ for $A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$, $m < M$, and $\varphi \in row_m[i]$ for $0 \leq i < |row_m|$. At the j-th iteration of line 4., row_j is non-deterministically generated, and it is checked whether $row_{j-1} \Rightarrow^{row_\varphi} row_j$. The procedure terminates after at most M iterations, where M is the maximum possible length of a simple path in $G_{\varphi\sim}$.

The working space used by the procedure is polynomial: M and *step* (which ranges in $[0, M-1]$) can be encoded in binary with $\lceil \log_2 M \rceil + 1 = O(|\varphi|^2)$ bits. At each step, we need to keep track of two φ-rows at a time, the current one, *row*, and its successor, *row*$'$: each φ-row can be represented as a sequence of at most $2|\varphi|$ (distinct) atoms, each one with an exponent that, by construction, cannot exceed M. Moreover, each φ-atom A can be represented using exactly $|\varphi|$ bits (for each $\psi \in \mathrm{CL}(\varphi)$, we set a bit to 1 if $\psi \in A$, and to 0 if $\neg\psi \in A$). Hence a φ-row can be encoded using $2|\varphi| \cdot (|\varphi| + \lceil \log_2 M \rceil + 1) = O(|\varphi|^3)$ bits. Finally, the condition $row \Rightarrow^{row_\varphi} row'$ can be checked by $O(|\varphi|^2)$ bits of space once we have guessed *row*$'$. This analysis entails the following result (we recall that **NPSPACE** = **PSPACE**).

▶ **Theorem 24.** *The satisfiability problem for* $\mathrm{D}|_{\mathcal{H}om}$*-formulas over finite linear orders is in* ***PSPACE****.*

We now outline which are the modifications to the previous concepts needed for proving the decidability of satisfiability for $\mathrm{D}|_{\mathcal{H}om}$ with the strict relation $\boxdot$, in place of $\sqsubset$. It is sufficient to replace the definitions of $\Rightarrow^{D_\varphi}$, φ-*row* and $\Rightarrow^{row_\varphi}$ with the following ones. For the sake of simplicity, we introduce a dummy atom $\boxdot$, for which we assume $\mathcal{R}eq_D(\boxdot) = \mathcal{O}bs_D(\boxdot) = \emptyset$.

▶ **Definition 25.** Given $A_1, A_3, A_4 \in \mathcal{A}_\varphi$ and $A_2 \in \mathcal{A}_\varphi \cup \{\boxdot\}$, we say that A_4 is $D_\varphi\, \boxdot$-generated by A_1, A_2, A_3, written $A_1, A_2, A_3 \Rightarrow^{D_\varphi \boxdot} A_4$ iff (i) $A_4 \cap \mathcal{AP} = A_1 \cap A_3 \cap \mathcal{AP}$ and (ii) $\mathcal{R}eq_D(A_4) = \mathcal{R}eq_D(A_1) \cup \mathcal{R}eq_D(A_3) \cup \mathcal{O}bs_D(A_2)$.

The idea of this definition is that, if an interval $[x, y]$, with $x < y$, is labeled by A_4, and the three subintervals $[x, y-1]$, $[x+1, y-1]$, and $[x+1, y]$ by A_1, A_2, A_3, resp., we want $A_1, A_2, A_3 \Rightarrow^{D_\varphi \boxdot} A_4$. In particular, if $x = y - 1$, then $A_2 = \boxdot$ (because $[x+1, y-1]$ is not a valid interval). Note that only $[x+1, y-1] \boxdot [x, y]$, hence we want $\mathcal{O}bs_D(A_2) \subseteq \mathcal{R}eq_D(A_4)$; moreover, since the requests of A_1 and A_3 are fulfilled by a strict subinterval of $[x, y]$, it must be $\mathcal{R}eq_D(A_1) \subseteq \mathcal{R}eq_D(A_4)$ and $\mathcal{R}eq_D(A_3) \subseteq \mathcal{R}eq_D(A_4)$.

▶ **Definition 26.** A φ-$\boxdot$-row is a finite sequence of φ-atoms $row = A_0^{m_0} \cdots A_n^{m_n}$ such that for every $0 \leq i \leq n$ we have $m_i > 0$, and for every $0 \leq j < i$, $\mathcal{R}eq_D(A_j) \subseteq \mathcal{R}eq_D(A_i)$, $A_i \neq A_j$, and $(A_j \cap \mathcal{AP}) \supseteq (A_i \cap \mathcal{AP})$. Moreover $\mathcal{R}eq_D(A_0) = \emptyset$.

▶ **Definition 27.** Given two φ-rows *row* and *row*$'$, we say that *row*$'$ is a successor of *row*, denoted as $row \Rightarrow^{row_\varphi \boxdot} row'$, if $|row'| = |row| + 1$, and for every $0 \leq i < |row|$, $row[i]row[i-1]row'[i] \Rightarrow^{D_\varphi \boxdot} row'[i+1]$, where we assume $row[i-1] = \boxdot$ if $i = 0$.

We conclude the section by stating the **PSPACE**-completeness of satisfiability for $\mathrm{D}|_{\mathcal{H}om}$ over finite linear orders (under both the *strict* and the *proper* semantic variants). The hardness proof can be found in [3].

▶ **Theorem 28.** *The satisfiability problem for* $\mathrm{D}|_{\mathcal{H}om}$*-formulas over finite linear orders is* ***PSPACE****-complete.*

4 Model checking for $\mathrm{D}|_{\mathcal{H}om}$ over Kripke structures

In this section we focus our attention on the *model checking (MC)* problem for $\mathrm{D}|_{\mathcal{H}om}$, namely, the problem of checking whether some behavioural properties, expressed as $\mathrm{D}|_{\mathcal{H}om}$-formulas,

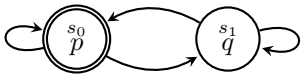


Figure 5 Kripke structure $\mathcal{K}_2$.

are satisfied by a model of a given system. The typical models are *Kripke structures*, which will now be introduced along with the semantic definition of $\mathrm{D}|_{\mathcal{H}om}$ over them.

▶ **Definition 29.** A *finite Kripke structure* is a tuple $\mathcal{K} = (\mathcal{AP}, W, E, \mu, s_0)$, where $\mathcal{AP}$ is a finite set of proposition letters, W is a finite set of states, $E \subseteq W \times W$ is a left-total relation between states, $\mu : W \to 2^{\mathcal{AP}}$ is a total labelling function, and $s_0 \in W$ is the initial state.

For all $s \in W$, $\mu(s)$ is the set of proposition letters that hold on s, while E is the transition relation that describes the evolution of the system over time.

Figure 5 depicts the finite Kripke structure $\mathcal{K}_2 = (\{p, q\}, \{s_0, s_1\}, E, \mu, s_0)$, with $E = \{(s_0,s_0),(s_0,s_1),(s_1,s_0),(s_1,s_1)\}$, $\mu(s_0) = \{p\}$, and $\mu(s_1) = \{q\}$. The initial state s_0 is identified by a double circle.

▶ **Definition 30.** A *trace* ρ of a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, W, E, \mu, s_0)$ is a finite sequence of states $s_1 \cdots s_n$, with $n \geq 1$, such that $(s_i, s_{i+1}) \in E$ for $i = 1, \ldots, n-1$.

For any trace $\rho = s_1 \cdots s_n$, we define: (i) $|\rho| = n$, and for $0 \leq i \leq |\rho| - 1$, $\rho(i) = s_{i+1}$; (ii) $\rho(i, j) = s_{i+1} \cdots s_{j+1}$, for $0 \leq i \leq j \leq |\rho| - 1$, is the subtrace of ρ bounded by i and j. Finally, if the first state of ρ is s_0 (the initial state of $\mathcal{K}$), ρ is called an *initial trace*.

▶ **Definition 31.** The interval model $\mathbf{M}_\rho = \langle \mathbb{I}(\mathbb{S}), \circ, \mathcal{V} \rangle$ *induced by a trace* ρ of a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, W, E, \mu, s_0)$ is the homogeneous interval model such that:
(i) $S = \{0, \ldots, |\rho| - 1\}$, and (ii) for all $x \in S$ and $p \in \mathcal{AP}$: $[x, x] \in \mathcal{V}(p)$ iff $p \in \mu(\rho(x))$.

▶ **Definition 32.** Let $\mathcal{K}$ be a finite Kripke structure and ψ be a $\mathrm{D}|_{\mathcal{H}om}$-formula. We say that a trace $\rho(i, j)$ of $\mathcal{K}$ satisfies ψ, denoted as $\mathcal{K}, \rho(i, j) \models \psi$, iff $\mathbf{M}_\rho, [i, j] \models \psi$. Moreover, we say that $\mathcal{K}$ models ψ, written $\mathcal{K} \models \psi$, iff for all *initial* traces ρ' of $\mathcal{K}$, it holds that $\mathcal{K}, \rho' \models \psi$. The *MC problem* for $\mathrm{D}|_{\mathcal{H}om}$ over finite Kripke structures is the problem of deciding if $\mathcal{K} \models \psi$.

Note that $p \in \mathcal{AP}$ holds over $\rho = s_1 \cdots s_n$ iff it holds over all the states $s_1, \ldots, s_n$ of ρ (*homogeneity assumption*). Since the number of initial traces of $\mathcal{K}$ is infinite, MC for $\mathrm{D}|_{\mathcal{H}om}$ over Kripke structures is not trivially decidable. We now describe how, with a slight modification of the previous satisfiability procedure, it is possible to derive a MC algorithm for $\mathrm{D}|_{\mathcal{H}om}$-formulas φ over finite Kripke structures $\mathcal{K}$. The idea is to consider some finite linear orders – not all the possible ones, unlike the case of satisfiability – precisely those corresponding to (some) initial traces of $\mathcal{K}$, checking whether $\neg\varphi$ holds over them: in such a case we have found a counterexample, and we can conclude that $\mathcal{K} \not\models \varphi$. To ensure this kind of "satisfiability driven by the traces of $\mathcal{K}$", we make a product between $\mathcal{K}$ and the previous graph $G_{\varphi\sim}$, getting what we call a "$(\varphi \sim \mathcal{K})$-*graph*". In the following, we will also exploit the notion of "compass structure *induced by a trace* ρ of $\mathcal{K}$", which is a fulfilling homogeneous compass φ-structure built from ρ and completely determined by it.

Given a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, W, E, \mu, s_0)$ and a $\mathrm{D}|_{\mathcal{H}om}$-formula φ, we consider the $(\varphi \sim \mathcal{K})$-graph $G_{\varphi\sim\mathcal{K}}$, which is basically the product of $\mathcal{K}$ and $G_{\varphi\sim} = (\mathcal{R}ows^{\sim}_{\varphi}, {}^{=row_\varphi}\!\!\rightarrow)$, formally defined as: $G_{\varphi\sim\mathcal{K}} = (\Gamma, \Xi)$, where:

- Γ is the maximal subset of $W \times \mathcal{R}ows^{\sim}_{\varphi}$ s.t.: if $(s, [row]_\sim) \in \Gamma$ then $\mu(s) = row[0] \cap \mathcal{AP}$;
- $\big((s_1, [row_1]_\sim), (s_2, [row_2]_\sim)\big) \in \Xi$ iff (i) $\big((s_1, [row_1]_\sim), (s_2, [row_2]_\sim)\big) \in \Gamma^2$, (ii) $(s_1, s_2) \in E$, and (iii) $[row_1]_\sim \, {}^{=row_\varphi}\!\!\rightarrow [row_2]_\sim$.

Input: a Kripke structure $\mathcal{K} = (\mathcal{AP}, W, E, \mu, s_0)$, a D$|_{\mathcal{H}om}$-formula φ

1. Put $M \leftarrow |W| \cdot 2^{3|\varphi|^2}$, $step \leftarrow 0$ and $(s, row) \leftarrow (s_0, A)$ for some atom $A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$ and $A \cap \mathcal{AP} = \mu(s_0)$.
2. If $\varphi \notin row[|row| - 1]$, return **yes**.
3. If $step = M - 1$, return **no**.
4. Non-deterministically choose s' such that $(s, s') \in E$.
5. Non-det. generate a φ-row row' and check that $row'[0] \cap \mathcal{AP} = \mu(s')$ and $row \,{}^{=row_\varphi}\!\!\rightarrow row'$.
6. Put $step \leftarrow step + 1$ and $(s, row) \leftarrow (s', row')$.
7. Go back to 2.

Figure 6 Non-deterministic procedure deciding the existence of initial traces ρ such that $\mathcal{K}, \rho \not\models \varphi$.

Note that the definition of Γ is well-given, since for all $row' \in [row]_\sim$, $row'[0] = row[0]$. The size of $G_{\varphi\sim\mathcal{K}}$ is bounded by $(|W| \cdot |\mathcal{R}ows^{\sim}_{\varphi}|)^2$.

Given a generic trace ρ of $\mathcal{K}$, we define the compass φ-structure *induced by* ρ as the fulfilling homogeneous compass φ-structure $\mathcal{G}_{(\mathcal{K},\rho)} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, where $S = \{0, \ldots, |\rho| - 1\}$, and for $0 \le x < |\rho|$, $\mathcal{L}(x, x) \cap \mathcal{AP} = \mu(\rho(x))$ and $\mathcal{R}eq_D(\mathcal{L}(x, x)) = \emptyset$. Note that, given ρ, $\mathcal{G}_{(\mathcal{K},\rho)}$ always exists and is unique: all φ-atoms $\mathcal{L}(x, x)$ "on the diagonal" are determined by the labeling of $\rho(x)$ (and by the absence of requests). Moreover, by Lemma 17, all the other atoms $\mathcal{L}(x, y)$, for $0 \le x < y < |\rho|$, are determined by the ${}^{=row_\varphi}\!\!\rightarrow$ relation between φ-rows.

The following property can easily be proved by induction.

▶ **Proposition 33.** *Given a Kripke structure $\mathcal{K}$, a trace ρ of $\mathcal{K}$, and a* D$|_{\mathcal{H}om}$*-formula φ, for all $0 \le x \le y < |\rho|$ and for all subformulas ψ of φ: $\mathcal{K}, \rho(x, y) \models \psi$ iff $\psi \in \mathcal{L}(x, y)$ in $\mathcal{G}_{(\mathcal{K},\rho)}$.*

We can now introduce Theorem 34, that can be regarded as a version of Theorem 23 for MC.

▶ **Theorem 34.** *Given a Kripke structure $\mathcal{K} = (\mathcal{AP}, W, E, \mu, s_0)$ and a* D$|_{\mathcal{H}om}$*-formula φ, there exists an initial trace ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \varphi$ iff there exists a path in $G_{\varphi\sim\mathcal{K}} = (\Gamma, \Xi)$ from some node $(s_0, [row]_\sim) \in \Gamma$ to some node $(s, [row']_\sim) \in \Gamma$ such that: (1) there is $row_1 \in [row]_\sim$ with $|row_1| = 1$, and (2) there is $row_2 \in [row']_\sim$ with $\varphi \in row_2[|row_2| - 1]$.*

Now, analogously to the case of satisfiability, we can perform a reachability in $G_{\varphi\sim\mathcal{K}}$, exploiting the previous theorem to decide whether there is an initial trace ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \neg\varphi$, for a D$|_{\mathcal{H}om}$-formula φ (i.e., the complementary problem of MC $\mathcal{K} \models \varphi$). The *non-deterministic* procedure of Figure 6 searches for a suitable path in $G_{\varphi\sim\mathcal{K}}$, $(s_0, [row_0]_\sim) \stackrel{\Xi}{\rightarrow} \cdots \stackrel{\Xi}{\rightarrow} (s_m, [row_m]_\sim)$, where $row_0 = A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$, $A \cap \mathcal{AP} = \mu(s_0)$, $m < M$, and $\neg\varphi \in row_m[|row_m| - 1]$ (i.e., $\varphi \notin row_m[|row_m| - 1]$). At the j-th iteration of lines 4./5., $(s_{j-1}, s_j) \in E$ is selected, and row_j is non-deterministically generated checking that $row_j[0] \cap \mathcal{AP} = \mu(s_j)$ and $row_{j-1} \,{}^{=row_\varphi}\!\!\rightarrow row_j$.

Basically, the same observations about the working space of the procedure in Figure 4 can be done also for this algorithm, except for the space used to encode in binary $M \le |W| \cdot 2^{3|\varphi|^2}$ and *step*, ranging in $[0, M - 1]$, which is $O(\log |W| + |\varphi|^2)$ bits. Moreover we need to store two states, s and s' of $\mathcal{K}$, that need $O(\log |W|)$ bits to be represented.

▶ **Theorem 35.** *The MC problem for* D$|_{\mathcal{H}om}$*-formulas over finite Kripke structures is **PSPACE**-complete. Moreover, for constant-length formulas, it is **NLOGSPACE**-complete.*

Proof. Membership is immediate by the previous space analysis, and the fact that the complexity classes **NPSPACE** = **PSPACE** and **NLOGSPACE** are closed under complement.

As for the **PSPACE**-hardness, we make a reduction from the **PSPACE**-complete *problem of universality* of the language of an NFA [8]. The full proof can be found in [3]. For the **NLOGSPACE**-hardness, there exists a trivial reduction from the *problem of (non-)reachability* of two nodes in a directed graph. ◀

Finally, it is possible to adapt the procedure also for *strict* $\mathrm{D}|_{\mathcal{H}om}$ (by exploiting Definitions 25–27).

5 Conclusions

In this paper, we have shown that both satisfiability and model checking for the logic D of sub-intervals – over finite linear orders and finite Kripke structures, respectively – are **PSPACE**-complete, under the homogeneity assumption. We are investigating the possibility of generalizing the given procedures to cope with the logic BE: nothing is known about its satisfiability, while a large gap separates known upper and lower bounds for model checking.

Acknowledgements. We sincerely thank an anonymous reviewer for his/her thorough review and valuable comments, which significantly contributed to improving the quality of the publication – in particular for spotting a problem with the hardness proof of satisfiability in the submitted paper, and suggesting a possible solution.

References

1 J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.

2 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Interval temporal logic model checking: The border between good and bad HS fragments. In *Proceedings of the 8th International Joint Conference (IJCAR)*, pages 389–405, 2016. doi:10.1007/978-3-319-40229-1_27.

3 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Satisfiability and model checking for the logic of sub-intervals under the homogeneity assumption. Technical report, University of Udine, Italy, 2017. URL: www.dimi.uniud.it/la-ricerca/pubblicazioni/preprints/4.2017/.

4 D. Bresolin, V. Goranko, A. Montanari, and P. Sala. Tableaux for logics of subinterval structures over dense orderings. *Journal of Logic and Computation*, 20(1):133–166, 2010. doi:10.1093/logcom/exn063.

5 D. Bresolin, V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighborhood logics: Expressiveness, decidability, and undecidable extensions. *Annals of Pure and Applied Logic*, 161(3):289–304, 2009. doi:10.1016/j.apal.2009.07.003.

6 D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. Interval temporal logics: a journey. *Bulletin of the EATCS*, 105:73–99, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/98.

7 J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38:279–292, 1991. doi:10.1145/115234.115351.

8 M. Holzer and M. Kutrib. Descriptional and computational complexity of finite automata – a survey. *Information and Computation*, 209(3):456–470, 2011. doi:10.1016/j.ic.2010.11.013.

9 H. Kamp and U. Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory, Volume 42 of Studies in Linguistics and Philosophy*. Springer, 1993.

10 K. Lodaya. Sharpening the undecidability of interval temporal logic. In *Proceedings of the 6th Asian Computing Science Conference (ASIAN)*, pages 290–298, 2000. doi:10.1007/3-540-44464-5_21.

11 J. Marcinkowski and J. Michaliszyn. The undecidability of the logic of subintervals. *Fundamenta Informaticae*, 131(2):217–240, 2014. doi:10.3233/FI-2014-1011.

12 A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron. Checking interval properties of computations. *Acta Informatica*, 53(6-8):587–619, 2016. doi:10.1007/s00236-015-0250-1.

13 A. Montanari. Interval temporal logics model checking. In *Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning, (TIME)*, page 2, 2016. doi:10.1109/TIME.2016.32.

14 A. Montanari, I. Pratt-Hartmann, and P. Sala. Decidability of the logics of the reflexive sub-interval and super-interval relations over finite linear orders. In *Proceedings of the 17th International Symposium on Temporal Representation and Reasoning (TIME)*, pages 27–34, 2010. doi:10.1109/TIME.2010.18.

15 M. Otto. Two variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66(2):685–702, 2001. doi:10.2307/2695037.

16 I. Shapirovsky. On PSPACE-decidability in transitive modal logic. In *Proceedings of the 5th conference on Advances in Modal logic (AiML)*, pages 269–287, 2004. URL: http://www.aiml.net/volumes/volume5/Shapirovsky.ps.

17 Y. Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990. doi:10.1305/ndjfl/1093635589.

18 Y. Venema. A modal logic for chopping intervals. *Journal of Logic and Computation*, 1(4):453–476, 1991. doi:10.1093/logcom/1.4.453.

Threshold Constraints with Guarantees for Parity Objectives in Markov Decision Processes*†

Raphaël Berthon[1], Mickael Randour[2], and Jean-François Raskin[3]

1 **ENS Rennes, Rennes, France**
`raphael.berthon@ens-rennes.fr`
2 **Computer Science Department, ULB – Université libre de Bruxelles, Brussels, Belgium**
`mickael.randour@gmail.com`
3 **Computer Science Department, ULB – Université libre de Bruxelles, Brussels, Belgium**
`jraskin@ulb.ac.be`

Abstract

The *beyond worst-case synthesis problem* was introduced recently by Bruyère et al. [10]: it aims at building system controllers that provide strict worst-case performance guarantees against an antagonistic environment while ensuring higher expected performance against a stochastic model of the environment. Our work extends the framework of [10] and follow-up papers, which focused on quantitative objectives, by addressing the case of ω-regular conditions encoded as parity objectives, a natural way to represent functional requirements of systems.

We build strategies that satisfy a main parity objective on all plays, while ensuring a secondary one with sufficient probability. This setting raises new challenges in comparison to quantitative objectives, as one cannot easily mix different strategies without endangering the functional properties of the system. We establish that, for all variants of this problem, deciding the existence of a strategy lies in $\mathsf{NP} \cap \mathsf{coNP}$, the same complexity class as classical parity games. Hence, our framework provides additional modeling power while staying in the same complexity class.

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases Markov decision processes, parity objectives, beyond worst-case synthesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.121

1 Introduction

Beyond worst-case synthesis. *Two-player zero-sum games* [18, 21] and *Markov decision processes* (MDPs) [17, 4] are popular frameworks for decision making in adversarial and uncertain environments respectively. In the former, a system controller (player 1) and its environment (player 2) compete antagonistically, and synthesis aims at strategies that ensure a specified behavior *against all possible strategies of the environment*. In the latter, the system is faced with a given stochastic model of its environment, and the focus is on satisfying a given level of expected performance, or a *specified behavior with a sufficient probability*.

The *beyond worst-case synthesis* framework [10] unites both views: we look for strategies that provide both *strict worst-case guarantees* and a *good level of performance against the*

* Full version is available on arXiv [5], `http://arxiv.org/abs/1702.05472`.

† Work partially supported by the ERC Starting grant 279499 (inVEST) and the ARC project "Non-Zero Sum Game Graphs: Applications to Reactive Synthesis and Beyond" (Fédération Wallonie-Bruxelles). J.-F. Raskin is Professeur Francqui de Recherche, M. Randour is an F.R.S.-FNRS postdoctoral researcher.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 121; pp. 121:1–121:15

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

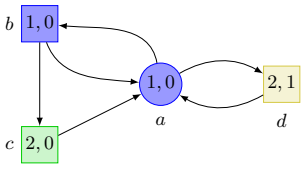


Figure 1 An MDP where player 1 can ensure p_1 surely and p_2 almost-surely.

stochastic model. Such requirements are natural in practical situations (e.g., see [9, 23] for applications to the shortest path problem). The original paper [10] dealt with mean-payoff and shortest path objectives. Follow-up work include, e.g., multi-dimensional extensions [14], optimization of the expected mean-payoff under hard Boolean constraints [1] or under energy constraints [7], or integration of beyond worst-case concepts in the tool Uppaal [15].

Parity objectives. We study the beyond worst-case problem for ω-regular conditions encoded as *parity objectives*. Parity games have been under close scrutiny for a long time due to their importance (e.g., they subsume modal μ-calculus model checking [16]) and their intriguing complexity: they belong to the class of problems in $\mathsf{NP} \cap \mathsf{coNP}$ [19] and despite many efforts (see [11] for pointers), whether they belong to P is still an open question.

In the aforementioned papers dealing with beyond worst-case problems, the focus was on *quantitative* objectives (e.g., mean-payoff). While it is usually the case that *qualitative* objectives, such as parity, are easier to deal with than quantitative ones, this is not true in the setting considered in this paper. Indeed, in the context of quantitative objectives, it is conceivable to alternate between two strategies along a play, such that one – efficient – strategy balances the performance loss due to playing the other – less efficient – strategy for a limited stretch of play infinitely often. In the context of qualitative objectives, this is no more possible in general, as one strategy may induce behaviors (such as invalidating the parity condition infinitely often) that can never be counteracted by the other one. Hence, in comparison, we need to define more elaborate analysis techniques to detect when satisfying *both* the worst-case and the probabilistic constraints with a single strategy is actually possible.

Example. Consider the MDP of Figure 1. Circle states are owned by player 1 (system) and square states are owned by player 2 (environment). In the stochastic model of the environment, square states are probabilistic, and, when not specified, we consider the uniform distribution over their successors. Each state is labelled with a name and two integers x, y representing priorities defined by two functions, p_1 and p_2. An infinite path in the graph is winning for player 1 and parity objective p_i, $i \in \{1, 2\}$, if the *maximal* priority seen infinitely often along the path for function p_i is *even*. We claim that player 1 has a strategy λ to ensure that (i) all plays consistent with λ satisfy p_1 (i.e., p_1 is surely satisfied) and (ii) the probability measure induced by λ on this MDP ensures that p_2 is satisfied with probability one (i.e., almost-surely).

One such λ is as follows. It plays an infinite sequence of rounds of n_i steps, $i \in \mathbb{N}$. In round i, in state a, the strategy chooses b for n_i steps, such that the probability to reach c during round i is larger than $1 - 2^{-i}$ (this is possible as at each step c is reached from b with probability $\frac{1}{2}$). If during round i, c is not reached (which can happen with a small probability) then λ goes to d once. Then the next round $i+1$ is started. This infinite-memory strategy ensures both (i) and (ii). Indeed, it can be shown that the probability that λ plays d infinitely often is zero. Also, during each round, the maximal priority for p_1 is guaranteed to be even because if c is not visited, d is systematically played.

Finally, we can prove that player 1 needs infinite memory to ensure p_1 surely and p_2 almost-surely, and also, that this is the best that player 1 can do here: he has no strategy to enforce surely both p_1 and p_2 at the same time.

Outline and contributions. We consider MDPs with two parity objectives (i.e., using different priority functions). We study the problem of deciding the existence of a strategy that ensures the first parity objective *surely* (i.e., on all plays) while yielding a *probability at least equal to (resp. greater than) a given rational threshold* to satisfy the second parity objective. In Section 2, we formally define the framework and recall important results from the literature. In Section 3, as an intermediate step, we solve the problem of ensuring the first parity objective surely while visiting a target set of states with sufficient probability: this tool will help us several times later. We prove that the corresponding decision problem is in $\mathsf{NP} \cap \mathsf{coNP}$ and at least as hard as parity games, and that finite-memory strategies are sufficient. In Section 4, we solve the problem for the two parity objectives, where the second one must hold *almost-surely* (i.e., with probability one). Our main tools are the novel notion of *ultra-good end-components*, as well as the reachability problem solved in Section 3. We generalize our approach to arbitrary probability thresholds in Section 5, in which we introduce the notion of *very-good end-components*. In both the almost-sure and the arbitrary threshold cases, we prove that the decision problem belongs to $\mathsf{NP} \cap \mathsf{coNP}$ and is at least as hard as parity games. In contrast to the reachability case, we prove that infinite memory is in general necessary. Full proofs are presented in the extended version of this paper [5].

Additional related work. The beyond worst-case synthesis framework illustrates the usefulness of non-zero-sum games for reactive synthesis [8, 22]. Other types of multi-objective specifications in stochastic models have been considered: e.g., percentile queries generalize the classical threshold probability problem to several dimensions [24]. In [3], Baier et al. study the quantitative analysis of MDPs under weak and strong fairness constraints. They provide algorithms for computing the probability for ω-regular properties in worst and best-case scenarios, when considering strategies that in addition satisfy weak or strong fairness constraints almost-surely. In contrast, we are able to consider similar objectives but for strategies that satisfy weak or strong fairness constraints surely, i.e., with certainty and not only with probability one. In [1], Almagor et al. consider the optimization of the expected mean-payoff under hard Boolean constraints in weighted MDPs. Our concept of ultra-good end-component builds upon their notion of super-good one. A reduction to mean-payoff parity games [12] is part of the identification process of both types of end-components.

2 Preliminaries

Directed graphs. A *directed graph* is a pair $G = (S, E)$ with S a set of vertices, called *states*, and $E \subseteq S \times S$ a set of directed edges. We focus here on *finite* graphs (i.e., $|S| < \infty$). Given a state $s \in S$, we denote by $\mathtt{Succ}(s) = \{s' \in S \mid (s, s') \in E\}$ the set of successors of s by edges in E. We assume that graphs are non-blocking, i.e., for all $s \in S$, $\mathtt{Succ}(s) \neq \emptyset$.

A *play* in G from an initial state $s \in S$ is an infinite sequence of states $\pi = s_0 s_1 s_2 \dots$ such that $s_0 = s$ and $(s_i, s_{i+1}) \in E$ for all $i \geq 0$. The *prefix* up to the $(n+1)$-th state of π is the finite sequence $\pi(0, n) = s_0 s_1 \dots s_n$. We resp. denote the first and last states of a prefix $\rho = s_0 s_1 \dots s_n$ by $\mathtt{First}(\rho) = s_0$ and $\mathtt{Last}(\rho) = s_n$. For a play π, we naturally extend the notation to $\mathtt{First}(\pi)$. Finally, for $i \in \mathbb{N}$, $\pi(i) = s_i$, and for $j > i$, $\pi(i, j) = s_i \dots s_j$. The set of plays of G is $\mathtt{Plays}(G)$ and the set of prefixes is $\mathtt{Pref}(G)$. For a set of plays Π, we

denote by $\texttt{Pref}(\Pi)$ the set of prefixes of these plays. Given two prefixes $\rho = s_0 \dots s_m$ and $\rho' = s'_0 \dots s'_n$ in $\texttt{Pref}(G)$, we denote their *concatenation* as $\rho \cdot \rho' = s_0 \dots s_m s'_0 \dots s'_n$. This is not necessarily a valid prefix of G. The same holds for a prefix concatenated with a play.

Probability distributions. Given a countable set A, a (rational) *probability distribution* on A is a function $p\colon A \to [0,1] \cap \mathbb{Q}$ such that $\sum_{a \in A} p(a) = 1$. We write $\mathcal{D}(A)$ the set of probability distributions on A. The *support* of $p \in \mathcal{D}(A)$ is $\texttt{Supp}(p) = \{a \in A \mid p(a) > 0\}$.

Markov decision processes. An *MDP* is a tuple $\mathcal{M} = (G, S_1, S_2, \delta)$ where (i) $G = (S, E)$ is a directed graph; (ii) (S_1, S_2) is a partition of S into states of player 1 (denoted by $\mathcal{P}_1$ and representing the system) and states of player 2 (denoted by $\mathcal{P}_2$ and representing the *stochastic* environment); (iii) $\delta\colon S_2 \to \mathcal{D}(S)$ is the transition function that, given a stochastic state $s \in S_2$, defines the probability distribution $\delta(s)$ over the successors of s, such that for all $s \in S_2$, $\texttt{Supp}(\delta(s)) = \texttt{Succ}(s)$. An MDP where for all $s \in S_1$, $|\texttt{Succ}(s)| = 1$ is a fully-stochastic process called a *Markov chain* (MC). A prefix $\rho \in \texttt{Pref}(\mathcal{M})$ belongs to $\mathcal{P}_i$, $i \in \{1,2\}$, if $\texttt{Last}(\rho) \in S_i$. The set of prefixes that belong to $\mathcal{P}_i$ is denoted by $\texttt{Pref}_i(\mathcal{M})$.

Strategies. A *strategy* for $\mathcal{P}_1$ is a function $\lambda\colon \texttt{Pref}_1(\mathcal{M}) \to \mathcal{D}(S)$, such that for all $\rho \in \texttt{Pref}_1(\mathcal{M})$, we have $\texttt{Supp}(\lambda(\rho)) \subseteq \texttt{Succ}(\texttt{Last}(\rho))$. The set of all strategies in $\mathcal{M}$ is denoted by Λ. *Pure* strategies have their support equal to a singleton for all prefixes. We mention that a strategy is *randomized* to stress on the need for randomness in general.

A strategy λ for $\mathcal{P}_1$ can be encoded by a stochastic state machine with outputs, called *stochastic Moore machine*, $\texttt{M}$. A strategy λ is *finite-memory* if $\texttt{M}$ is finite, and *memoryless* if it has only one state. That is, it does not depend on the history but only on the current state of the MDP: in this case, we have that $\lambda\colon S_1 \to \mathcal{D}(S)$. Finally, if the same strategy can be used regardless of the initial state, we say that a *uniform* strategy exists.

A play π is *consistent* with a strategy λ if for all $n \geq 0$ such that $\pi(n) \in S_1$, we have that $\pi(n+1) \in \texttt{Supp}(\lambda(\pi(0,n))$. It is defined similarly for prefixes. We write $\texttt{Out}^{\mathcal{M}}(\lambda) \subseteq \texttt{Plays}(G)$ the set of plays consistent with λ. We use $\texttt{Out}^{\mathcal{M}}_s(\lambda)$ when fixing an initial state s.

Markov chain induced by a strategy. An MDP $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$ and a strategy λ for $\mathcal{P}_1$ determine an MC $\mathcal{C} = (G', \delta')$. Given $s \in S$ an initial state and $\mathcal{A} \subseteq \texttt{Plays}(G)$ a measurable set, we denote by $\mathbb{P}^{\lambda}_{\mathcal{M},s}[\mathcal{A}]$ the *probability* of event $\mathcal{A}$ when $\mathcal{M}$ is executed with initial state s and strategy λ.

Objectives. Given an MDP $\mathcal{M} = (G, S_1, S_1, \delta)$, an *objective* is a set of plays $\mathcal{A} \subseteq \texttt{Plays}(G)$. We consider two classical objectives from the literature. Both define measurable events. To define them, we introduce the following notation: given a play $\pi \in \texttt{Plays}(G)$, let $\inf(\pi) = \{s \in S \mid \forall\, i \geq 0, \exists\, j \geq i, \pi(j) = s\}$ be the set of states seen infinitely often along π.

Reachability. Given a target $T \subseteq S$, this objective asks for plays that visit T: $\texttt{Reach}(T) = \{\pi \in \texttt{Plays}(G) \mid \exists\, n \geq 0, \pi(n) \in T\}$. We later use the LTL notation $\Diamond T$ for event $\texttt{Reach}(T)$.

Parity. Let $p\colon S \to \{1, 2, \dots, d\}$ be a *priority function* that maps each state to an integer priority, where $d \leq |S| + 1$ (w.l.o.g.). The parity objective asks that, among the priorities seen infinitely often, the *maximal* one be even: $\texttt{Parity}(p) = \{\pi \in \texttt{Plays}(G) \mid \max_{s \in \inf(\pi)} p(s) \text{ is even}\}$. We later simply use p to denote the event $\texttt{Parity}(p)$.

End-components and sub-MDPs. Let $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$ be an MDP. An *end-component* (EC) of $\mathcal{M}$ is a set $C \subseteq S$ such that (i) $\forall\, s \in C \cap S_2$, $\texttt{Succ}(s) \subseteq C$ and $\forall\, s \in C \cap S_1$,

$\mathtt{Succ}(s) \cap C \neq \emptyset$; and (ii) C is *strongly connected*, i.e., for any two states $s, s' \in C$, there exists a path from s to s' that stays in C. It is well-known that inside an EC C, $\mathcal{P}_1$ can force the visit of any state $s \in C$ with probability 1 (that is, when $\mathcal{P}_2$ is seen as stochastic and obeys the strategy δ), see e.g., [4]. The union of two ECs with non-empty intersection is an EC. An EC C is thus *maximal* if, for every EC C', $C' \subseteq C \vee C' \cap C = \emptyset$.

Given an EC $C \subseteq S$ of $\mathcal{M}$, we write $\mathcal{M}_{\restriction C}$ the *sub-MDP* defined by $\mathcal{M}_{\restriction C} = (G' = (C, E \cap C \times C), S'_1 = S_1 \cap C, S'_2 = S_2 \cap C, \delta')$, where $\delta' \colon S'_2 \to \mathcal{D}(C)$ is the restriction of δ to the domain C. Note that $\mathcal{M}_{\restriction C}$ is a well-defined MDP: it has no deadlock since C is strongly connected and in all stochastic states s, $\mathtt{Supp}(\delta'(s)) \subseteq C$ (as C was an EC in $\mathcal{M}$).

Technical lemma. We recall a classical result about MDPs that will be useful later on.

▶ **Lemma 1** (Optimal reachability [4]). *Given an MDP $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$ and a target set $T \subseteq S$, we can compute for each state $s \in S$ the maximal probability $v^*_s = \sup_{\lambda \in \Lambda} \mathbb{P}^\lambda_{\mathcal{M},s}[\Diamond T]$ to reach T, in polynomial time. There is an optimal uniform pure memoryless strategy λ^* that enforces v^*_s from all $s \in S$. Now, fix $s \in S$ and $c \in \mathbb{Q}$ such that $c < v^*_s$. Then there exists $k \in \mathbb{N}$ such that by playing λ^* from s for k steps, we reach T with probability larger than c.*

Events and probabilistic operators. Consider an MDP $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$. Recall that we have defined two types of measurable events (specific subsets of $\mathtt{Plays}(G)$) with respective notations $\Diamond T$ for $T \subseteq S$ (reachability), and p for $p \colon S \to \{1, \dots, d\}$ a priority function (parity). We define three operators to reason about the probabilities of these events: $\mathtt{S}$, $\mathtt{P}_{\sim c}$, and $\mathtt{AS}$. Given an event $\mathcal{A}$ and a state s, they are used as follows:

- $\mathcal{A}$ is *sure* from s, denoted $s \models \mathtt{S}(\mathcal{A})$, if there exists a strategy λ of $\mathcal{P}_1$ such that $\mathtt{Out}^{\mathcal{M}}_s(\lambda) \subseteq \mathcal{A}$. Here probabilities are ignored and we consider $\mathcal{P}_2$ as antagonistic.
- $\mathcal{A}$ *holds with probability at least equal to (resp. greater than)* $c \in \mathbb{Q}$ from s, denoted $s \models \mathtt{P}_{\geq c}(\mathcal{A})$ (resp. $s \models \mathtt{P}_{>c}(\mathcal{A})$) if there exists λ such that $\mathbb{P}^\lambda_{\mathcal{M},s}[\mathcal{A}] \geq c$ (resp. $> c$).
- $\mathcal{A}$ is *almost-sure* from s, denoted $s \models \mathtt{AS}(\mathcal{A})$, if there exists λ such that $\mathbb{P}^\lambda_{\mathcal{M},s}[\mathcal{A}] = 1$.

For any operator $\mathtt{O}$, we say that such a λ is a *witness strategy* for $s \models \mathtt{O}(\mathcal{A})$ and we write $s, \lambda \models \mathtt{O}(\mathcal{A})$ to denote it. We will also consider *combinations* of the type $s \models \mathtt{O}_1(\mathcal{A}_1) \wedge \mathtt{O}_2(\mathcal{A}_2)$ for two operators and events: in this case, we require that the same strategy be a witness for both conjuncts, i.e., that there exists λ such that $s, \lambda \models \mathtt{O}_1(\mathcal{A}_1)$ and $s, \lambda \models \mathtt{O}_2(\mathcal{A}_2)$. Finally, we will sometimes use different MDPs, in which case we add the considered MDP $\mathcal{M}$ as a subscript on $\models$, e.g., $s \models_{\mathcal{M}} \mathtt{O}(\mathcal{A})$. We drop this subscript when the context is clear.

Beyond worst-case problems. Let $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$ be an MDP, $s \in S$ be an initial state, and p_1, p_2 be two priority functions. We provide algorithms to decide the existence of a witness strategy — and synthesize it — for the following formulae: (i) $s \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$, and (ii) $s \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\sim c}(p_2)$ for $\sim \in \{>, \geq\}$ and $c \in \mathbb{Q} \cap [0, 1)$.

3 Reachability under parity constraints

We study two variants, given $s \in S$, $T \subseteq S$, and $p \colon S \to \{1, \dots d\}$: (i) $s \models \mathtt{S}(p) \wedge \mathtt{AS}(\Diamond T)$, and (ii) $s \models \mathtt{S}(p) \wedge \mathtt{P}_{\sim c}(\Diamond T)$ for $\sim \in \{>, \geq\}$ and $c \in \mathbb{Q} \cap [0, 1)$.

Almost-sure reachability. This case can be solved by reduction to a slight variant studied in [2, Lemma 3] (extended version of [1]). The approach of [2, Lemma 3] relies on a reduction

to a Büchi-parity game: sufficiency of finite memory follows from this reduction. The lower complexity bound is trivial: it suffices to fix $T = S$ to obtain a classical parity game [19].

▶ **Theorem 2.** *Given an MDP $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$, a state $s_0 \in S$, a priority function $p\colon S \to \{1, \dots, d\}$, and a target set of states $T \subseteq S$, it can be decided in $\mathsf{NP} \cap \mathsf{coNP}$ if $s_0 \models \mathtt{S}(p) \wedge \mathtt{AS}(\Diamond T)$. If the answer is* Yes*, then there exists a finite-memory witness strategy. This decision problem is at least as hard as solving parity games.*

Reachability with threshold probability. We first study strategies that maximize the probability of reaching a target $T \subseteq S$ in an MDP $\mathcal{M}$. By Lemma 1, we have an optimal uniform pure memoryless strategy λ^* that enforces v^*_s from all $s \in S$. We define the set $E^{\neg\mathrm{opt}} = \{(s, s') \in E \mid s \in S_1 \wedge v^*_s > v^*_{s'}\}$ that contains all edges that are *non-optimal choices* for $\mathcal{P}_1$ in the sense that they result in a strict decrease of the probability to reach T. We show that playing, for a finite number of steps, edges that are optimal (i.e., in $E^{\mathrm{opt}} = E \setminus E^{\neg\mathrm{opt}}$), and then switching to an optimal strategy, like λ^*, produces an optimal strategy too.

▶ **Lemma 3.** *Let λ^* be an optimal uniform pure memoryless strategy in $\mathcal{M}$ to reach T, from all states in S. If λ is a strategy that plays only edges in E^{opt} for m steps, for $m \in \mathbb{N}$, and then switches to λ^*, then λ is also optimal to reach T from all states in S.*

We now turn to the problem $s_0 \models \mathtt{S}(p) \wedge \mathtt{P}_{\sim c}(\Diamond T)$ and establish the following result.

▶ **Theorem 4.** *Given an MDP $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$, a state $s_0 \in S$, a priority function $p\colon S \to \{1, \dots, d\}$, a target set of states $T \subseteq S$, and a probability threshold $c \in [0, 1) \cap \mathbb{Q}$, it can be decided in $\mathsf{NP} \cap \mathsf{coNP}$ if $s_0 \models \mathtt{S}(p) \wedge \mathtt{P}_{\sim c}(\Diamond T)$ for $\sim \in \{>, \geq\}$. If the answer is* Yes*, then there exists a finite-memory witness strategy. This decision problem is at least as hard as solving parity games.*

Proof Sketch. First, we restrict $\mathcal{M}$ to the $\subseteq$-maximal sub-MDP $\mathcal{M}^{\mathrm{w}}$ in which $\mathcal{P}_1$ can ensure $\mathtt{S}(p_1)$ from all states, by solving a classical parity game, which is in $\mathsf{NP} \cap \mathsf{coNP}$ [19]. Indeed, if $s_0 \not\models \mathtt{S}(p)$, the answer is No. In $\mathcal{M}^{\mathrm{w}}$, $\mathcal{P}_1$ has a uniform pure memoryless strategy λ^p that ensures $\mathtt{S}(p)$ from every state.

The case $> c$ is the easier. First, we compute the maximal probability $v^*_{s_0}$ to reach T and an optimal strategy λ^*, in polynomial time (Lemma 1). If $v^*_{s_0} \leq c$, then the answer is clearly No. Otherwise, we claim it is Yes. We construct a witness strategy λ for $s_0 \models \mathtt{S}(p) \wedge \mathtt{P}_{>c}(\Diamond T)$ from λ^* and λ^p as follows. Starting in s_0, the strategy λ plays as λ^* for k steps where k is taken as in Lemma 1: the probability to reach T after k steps is strictly greater than c, which implies that $s_0, \lambda \models \mathtt{P}_{>c}(\Diamond T)$. Then, λ switches to λ^p. Since parity is prefix-independent, we have that $s_0, \lambda \models \mathtt{S}(p)$, and we are done. Our procedure lies in $\mathsf{P}^{\mathsf{NP} \cap \mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$ [6], and λ is finite-memory since λ^* and λ^p are memoryless and k is finite.

We now turn to case $\geq c$. We compute $v^*_{s_0}$ in polynomial time. If $v^*_{s_0} > c$, then we answer Yes as we apply the same reasoning as in the previous case. If $v^*_{s_0} < c$, then we trivially answer No. The more involved case is $v^*_{s_0} = c$. We must verify that probability c is still achievable if, in addition, it is required to enforce $\mathtt{S}(p)$. To answer this, we modify $\mathcal{M}^{\mathrm{w}}$ and we reduce our problem to the almost-sure case of Theorem 2. Intuitively, we construct the MDP $\mathcal{M}'$ as follows. (i) We enrich states with one bit that records if T has been visited. (ii) While T has not been visited, we suppress all edges controlled by $\mathcal{P}_1$ that are not optimal for reachability, i.e., all edges in $E^{\neg\mathrm{opt}}$. (iii) While T has not been visited, we delete all states that cannot reach T and normalize the probability of the edges that survive this deletion.

We prove that $s'_0 \models_{\mathcal{M}'} \mathtt{S}(p) \wedge \mathtt{AS}(\Diamond T') \iff s_0 \models_{\mathcal{M}^{\mathrm{w}}} \mathtt{S}(p) \wedge \mathtt{P}_{\geq c}(\Diamond T)$ holds, where s'_0 is the initial state in $\mathcal{M}'$ and T' the translation of T. The crux is the restriction to E^{opt}

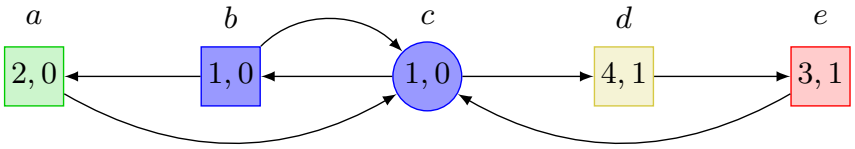


Figure 2 This MDP is a UGEC: going to d satisfies $(\mathbf{1_U})$, whereas going to b satisfies $(\mathbf{2_U})$.

before visiting T: $\mathcal{P}_1$ must be able to ensure $\mathtt{S}(p_1)$ while using only edges that are optimal for reachability if T cannot be forced surely. As the almost-sure problem is in $\mathsf{NP} \cap \mathsf{coNP}$ by Theorem 2, and $\mathcal{M}'$ is polynomially larger than $\mathcal{M}^{\mathrm{w}}$ (and thus $\mathcal{M}$), we obtain the claimed complexity using $\mathsf{P}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$ [6]. Our reduction implies that the witness strategy can be finite-memory. Again, this problem generalizes parity games by taking $T = S$. ◀

4 Almost-sure parity under parity constraints

Overview and key lemma. We consider an MDP $\mathcal{M} = (G = (S, E), S_1, S_1, \delta)$ with two priority functions p_1 and p_2. We look at the problem $s \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$. The cornerstone of our approach is the notion of *ultra-good end-component*.

▶ **Definition 5.** An end-component C of $\mathcal{M}$ is *ultra-good* (UGEC) if in the sub-MDP $\mathcal{M}_{\restriction C}$, the following two properties hold:

- $(\mathbf{1_U})$ $\forall\, s \in C$, $s \models_{\mathcal{M}_{\restriction C}} \mathtt{S}(p_1) \wedge \mathtt{AS}(\Diamond C^{\max}_{\mathrm{even}}(p_1))$, where

$$C^{\max}_{\mathrm{even}}(p_i) = \left\{ s \in C \mid (p_i(s) \text{ is even}) \wedge (\forall\, s' \in C,\ p_i(s') \text{ is odd} \implies p_i(s') < p_i(s)) \right\}$$

  contains the states with even priorities that are larger than any odd priority in C (this set can be empty for arbitrary ECs but needs to be non-empty for UGECs);
- $(\mathbf{2_U})$ $\forall\, s \in C$, $s \models_{\mathcal{M}_{\restriction C}} \mathtt{AS}(p_1) \wedge \mathtt{AS}(p_2)$, or equivalently, $s \models_{\mathcal{M}_{\restriction C}} \mathtt{AS}(p_1 \cap p_2)$.

We introduce the following notations: $\mathrm{UGEC}(\mathcal{M})$ is the set of all UGECs of $\mathcal{M}$, and $\mathcal{U} = \cup_{U \in \mathrm{UGEC}(\mathcal{M})} U$ is the set of states that belong to a UGEC in $\mathcal{M}$.

Intuitively, within a UGEC, $\mathcal{P}_1$ has a strategy to almost-surely visit $C^{\max}_{\mathrm{even}}(p_1)$ while guaranteeing $\mathtt{S}(p_1)$, and he also has a (generally different) strategy that almost-surely ensures both parity objectives. Figure 2 gives an example of UGEC. This notion strengthens the concept of *super-good* EC from [1]: essentially, the super-good ECs are exactly the ECs satisfying $(\mathbf{1_U})$. Thus, every UGEC is a super-good EC, but the converse is false.

The central lemma underpinning our approach is the following.

▶ **Lemma 6.** *The following equivalence holds:*

$$s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(\Diamond \mathcal{U}) \iff s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2).$$

Essentially, this lemma permits to reduce the problem under study to the one treated in Theorem 2, provided that we are able to compute $\mathcal{U}$, the set of states appearing in a UGEC. The rest of this section is dedicated to the proof of this lemma and its consequences.

Left-to-right implication (sufficient condition). We first study witness strategies for conditions $(\mathbf{1_U})$ and $(\mathbf{2_U})$ of Definition 5. For $(\mathbf{1_U})$, it was shown in the proof of [2, Lemma 3] (extended version of [1]) that deciding if the condition holds is in $\mathsf{NP} \cap \mathsf{coNP}$ and that uniform finite-memory witness strategies exist. For $(\mathbf{2_U})$, we establish the following lemma.

▶ **Lemma 7.** *Let C be an EC of $\mathcal{M}$. The following assertions hold.*

1. *It can be decided in polynomial time if condition $(\mathbf{2_U})$ holds.*
2. *If it holds, then there exists a (uniform randomized) memoryless witness strategy $\lambda_{2,C}$ and a sub-EC $D \subseteq C$ such that $D^{\max}_{\text{even}}(p_1) \neq \emptyset$, $D^{\max}_{\text{even}}(p_2) \neq \emptyset$, and for all $s \in C$, we have that $\mathbb{P}^{\lambda_{2,C}}_{\mathcal{M}_{\restriction C},s}\left[\{\pi \in \mathtt{Out}^{\mathcal{M}_{\restriction C}}(\lambda_{2,C}) \mid \inf(\pi) = D\}\right] = 1$.*
3. *Furthermore, $\lambda_{2,C}$ satisfies the following property: $\forall\, s \in C$, $\forall\, \varepsilon > 0$, $\exists\, n \in \mathbb{N}$ such that $\mathbb{P}^{\lambda_{2,C}}_{\mathcal{M}_{\restriction C},s}\left[\{\pi \in \mathtt{Out}^{\mathcal{M}_{\restriction C}}(\lambda_{2,C}) \mid \exists\, i,\, 0 \leq i \leq n,\, \pi(i) \in D^{\max}_{\text{even}}(p_1)\}\right] \geq 1 - \varepsilon$.*

Proof Sketch. For Point **2**, we resort on a classical result on almost-sure reachability of ECs and almost-sure satisfaction of both parity objectives. For Point **3**, we use Lemma 1 and Point **2**. For Point **1**, we show that (i) the existence of a sub-EC D such that $D^{\max}_{\text{even}}(p_1) \neq \emptyset$ and $D^{\max}_{\text{even}}(p_2) \neq \emptyset$ is not only necessary but also sufficient to satisfy condition $(\mathbf{2_U})$, and (ii) the existence of such a set can be decided in polynomial time. For (i), it suffices to build a uniform randomized memoryless strategy λ that reaches the sub-EC D almost-surely and then plays uniformly at random in it forever: λ will be a witness for $s \models_{\mathcal{M}_{\restriction C}} \mathtt{AS}(p_1) \wedge \mathtt{AS}(p_2)$, so condition $(\mathbf{2_U})$ holds in C. For (ii), we first check if $C^{\max}_{\text{even}}(p_1) \neq \emptyset$ and $C^{\max}_{\text{even}}(p_2) \neq \emptyset$. If this holds, then $D = C$ and the answer is Yes (it takes linear time obviously). If it does not hold, then we compute the sets $C^{\max}_{\text{odd}}(p_i) = \big\{s \in C \mid (p_i(s) \text{ is odd}) \wedge (\forall\, s' \in C,\, p_i(s') \text{ is even} \implies p_i(s') < p_i(s))\big\}$ and we iterate this procedure in the sub-EC $C' \subset C$ defined as $C' = C \setminus \mathtt{Attr}_2\big(C^{\max}_{\text{odd}}(p_1) \cup C^{\max}_{\text{odd}}(p_2)\big)$, where $\mathtt{Attr}_2$ is the classical attractor for $\mathcal{P}_2$. A suitable D exists if and only if this procedure stops before $C' = \emptyset$. In addition, this procedure takes at most $|C|$ iterations (as we remove at least one state at each step) and each iteration takes linear time. ◀

We will now prove that inside any UGEC, there is a strategy for $\mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$. From now on, let C be a UGEC of $\mathcal{M}$, $\lambda_{1,C}$ be a uniform finite-memory witness strategy for $(\mathbf{1_U})$ in Definition 5, and $\lambda_{2,C}$ be a uniform randomized memoryless one for $(\mathbf{2_U})$, additionally satisfying the properties of Lemma 7. We build a strategy λ_C based on $\lambda_{1,C}$ and $\lambda_{2,C}$.

▶ **Definition 8.** Let $C \in \mathrm{UGEC}(\mathcal{M})$. Let $(n_i)_{i\in\mathbb{N}}$ be a sequence of naturals n_i such that $\mathbb{P}^{\lambda_{2,C}}_{\mathcal{M}_{\restriction C},s}\left[\{\pi \in \mathtt{Out}^{\mathcal{M}_{\restriction C}}(\lambda_{2,C}) \mid \exists\, i,\, 0 \leq i \leq n_i,\, \pi(i) \in D^{\max}_{\text{even}}(p_1)\}\right] \geq 1 - 2^{-i}$, whose existence is guaranteed by Lemma 7. We build strategy λ_C as follows, starting with $i = 0$.

(a) Play $\lambda_{2,C}$ for n_i steps. Then $i = i + 1$ and go to **(b)**.

(b) If $D^{\max}_{\text{even}}(p_1)$ was visited in phase **a)**, then go to **(a)**.
Else, play $\lambda_{1,C}$ until $C^{\max}_{\text{even}}(p_1)$ is reached and then go to **(a)**.

Observe that λ_C requires infinite memory. In the next lemma, we prove that λ_C is a proper witness for $\mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$ in the UGEC C.

▶ **Lemma 9.** *Let $C \in \mathrm{UGEC}(\mathcal{M})$. For all $s \in C$, it holds that $s, \lambda_C \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$.*

Proof Sketch. First consider $s, \lambda_C \models \mathtt{S}(p_1)$. Fix any $\pi \in \mathtt{Out}^{\mathcal{M}_{\restriction C}}_s(\lambda_C)$: we will show that $\max_{s' \in \inf(\pi)} p_1(s')$ is even. Three cases are possible: (i) λ_C switches infinitely often between $\lambda_{1,C}$ and $\lambda_{2,C}$, (ii) it eventually plays $\lambda_{1,C}$ forever, and (iii) it eventually plays $\lambda_{2,C}$ forever. In case (i), $C^{\max}_{\text{even}}(p_1)$ is visited infinitely often. Since any state in this set has an even priority higher than any odd priority in C, we are good. In case (ii), we know that $s, \lambda_{1,C} \models \mathtt{S}(p_1)$. By prefix-independence, we are also good. In case (iii), $D^{\max}_{\text{even}}(p_1)$ is visited infinitely often, and, eventually, play π never leaves the sub-EC D. Hence, we are good here too and we conclude that $s, \lambda_C \models \mathtt{S}(p_1)$.

To show that $s, \lambda_C \models \mathtt{AS}(p_2)$, we prove that λ_C almost-surely ends up in playing only $\lambda_{2,C}$ (which ensures $\mathtt{AS}(p_2)$). The crux here is the choice of durations n_i for the strategy: we

can show that the probability to never play $\lambda_{1,C}$ again after round i tends to one when i tends to infinity. This entails the needed result. ◀

We can now prove the left-to-right implication of Lemma 6. For this, assume that for $s_0 \in S$, we have that $\lambda_{\mathcal{U}}$ is a witness for $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(\Diamond \mathcal{U})$, where we recall that $\mathcal{U}$ represents the union of all UGECs of the MDP $\mathcal{M}$. Note that such a strategy can be finite-memory w.l.o.g. as proved in Theorem 2. We build a global strategy λ as follows.

▶ **Definition 10.** Based on strategies $\lambda_{\mathcal{U}}$ and λ_C for all $C \in \mathrm{UGEC}(\mathcal{M})$, we build the global strategy λ as follows.
(a) Play $\lambda_{\mathcal{U}}$ until a UGEC C is reached, then go to **(b)**.
(b) Play λ_C forever.

This strategy requires infinite memory because of the strategies λ_C. We prove that λ is a witness for $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$.

▶ **Lemma 11.** *It holds that* $s_0, \lambda \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$.

Right-to-left implication (necessary condition). We now turn to the converse implication of Lemma 6, i.e., that $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$ implies $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(\Diamond \mathcal{U})$. We start by an intermediate lemma regarding witness strategies: it establishes that all states reachable via such a strategy also satisfy the property.

▶ **Lemma 12.** *For every state* $s \in S$*, every strategy* λ *such that* $s, \lambda \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$*, and every prefix* $\rho \in \mathtt{Pref}(\mathtt{Out}_s^{\mathcal{M}}(\lambda))$*, we have that* $\mathtt{Last}(\rho) \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$.

The next lemma establishes that at least one UGEC must exist in $\mathcal{M}$.

▶ **Lemma 13.** *The following holds:* $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2) \implies \mathrm{UGEC}(\mathcal{M}) \neq \emptyset$.

Proof Sketch. Given $\Pi \subseteq \mathtt{Plays}(G)$, we define $\mathtt{States}(\Pi) = \{s \in S \mid \exists \pi \in \Pi, \exists n \in \mathbb{N}, \pi(n) = s\}$. We then study the set

$$\mathcal{S} = \left\{ R \subseteq S \mid \exists\, s \in S, \exists \lambda \in \Lambda, (s, \lambda \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)) \wedge (R = \mathtt{States}(\mathtt{Out}_s^{\mathcal{M}}(\lambda))) \right\}.$$

Intuitively, it contains any subset of S that captures all states reachable by some witness strategy λ, from some state $s \in S$. First note that $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$ implies that $\mathcal{S}$ is non-empty, as for a witness strategy λ, $R = \mathtt{States}(\mathtt{Out}_{s_0}^{\mathcal{M}}(\lambda)) \in \mathcal{S}$, by definition.

We then show that all minimal elements of $\mathcal{S}$ for set inclusion $\subseteq$ are UGECs, which suffices to establish our lemma. The most important ingredients to prove that any $R \in \min_{\subseteq}(\mathcal{S})$ is a UGEC are the following. First the existence, for any $s \in R$, of a strategy λ_R such that $s, \lambda_R \models_{\mathcal{M}_{\restriction R}} \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$ (i.e., λ_R satisfies the property without leaving R), which follows from Lemma 12 and the minimality of R in $\mathcal{S}$. Second, proving that $R_{\mathrm{even}}^{\max}(p_1)$ is dense in the subtree induced by $\mathtt{Out}_s^{\mathcal{M}}(\lambda_R)$, that is, that for every prefix ρ, the subtree defined by λ_R from ρ reaches a state of $R_{\mathrm{even}}^{\max}(p_1)$, and that this holds in all subsequent subtrees. From this density argument, we can derive a witness strategy for condition $(\mathbf{1_U})$ in Definition 5. ◀

Collecting in $\mathcal{U}_{\min} = \cup_{R \in \min_{\subseteq}(\mathcal{S})} R$ all states that belong to minimal sets R of $\mathcal{S}$, we finally prove the implication.

▶ **Lemma 14.** *The following holds:* $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2) \implies s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(\Diamond \mathcal{U}_{\min})$.

Algorithm. Lemma 11 and Lemma 14 prove the correctness of the reduction presented in Lemma 6. It is the cornerstone of our algorithm.

▶ **Theorem 15.** *Given an MDP $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$, a state $s_0 \in S$, and two priority functions $p_i \colon S \to \{1, \dots, d\}$, $i \in \{1, 2\}$, it can be decided in* NP∩coNP *if $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$. If the answer is* Yes*, then there exists an infinite-memory witness strategy, and infinite memory is in general necessary. This decision problem is at least as hard as solving parity games.*

Proof. The algorithm can be sketched as follows:

1. Compute the set $\max_{\subseteq}(\mathrm{SGEC}(\mathcal{M}))$ of maximal super-good ECs, using [1]. Those are the maximal ECs satisfying condition $(\mathbf{1_U})$ in Definition 5. There are only polynomially many of them, and their computation is in NP ∩ coNP.
2. For each of them, check if $(\mathbf{2_U})$ holds using Lemma 7, in polynomial time. If an EC does not satisfy $(\mathbf{2_U})$, then it is also the case of all its sub-ECs (as seen in the proof of Lemma 7). Hence, we have that $\mathcal{U} = \{C \in \max_{\subseteq}(\mathrm{SGEC}(\mathcal{M})) \mid C \text{ satisfies } (\mathbf{2_U})\}$.
3. Decide if $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(\Diamond\mathcal{U})$ using Theorem 2. This is in NP ∩ coNP. If it holds, then answer Yes, otherwise answer No.

Its correctness was established in Lemma 6. It belongs to $\mathsf{P}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$ [6], and it trivially generalizes classical parity games (e.g., by taking $p_2 \colon s \mapsto 0$ for all $s \in S$).

Finally, let us discuss strategies. A witness strategy λ plays as follows: (i) it plays as the finite-memory strategy witness for $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(\Diamond\mathcal{U})$ given by Theorem 2 until a UGEC C is reached, (ii) then it switches to the infinite-memory strategy λ_C described in Definition 8. It is clear that such a strategy is a witness for $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{AS}(p_2)$, as expected.

Infinite memory is required in general, as shown in the UGEC C in Figure 2: there exists no finite-memory witness strategy in C. Indeed, assume $\mathcal{P}_1$ is restricted to a finite-memory strategy λ. To be able to ensure p_1 on the play in which $\mathcal{P}_2$ always goes to c from b, $\mathcal{P}_1$ must visit d infinitely often, and because of the finite memory of λ, he must do it after a bounded number of steps along which a is not visited: say n steps. Hence, the probability to do it will be bounded from below by a strictly positive constant, here $2^{-\frac{n}{2}}$ (the probability that $\mathcal{P}_2$ chooses c for $\frac{n}{2}$ times in a row), all along a consistent play. Therefore, $\mathcal{P}_1$ will almost-surely visit d infinitely often, and p_2 will actually be satisfied with probability zero. ◀

5 Parity with threshold probability under parity constraints

We now turn to the problem $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\sim c}(p_2)$ for $\sim \in \{>, \geq\}$ and $c \in \mathbb{Q} \cap [0, 1)$.

Very-good end-components. In addition to UGECs, we need the new notion of *very-good end-component*.

▶ **Definition 16.** An end-component C of $\mathcal{M}$ is *very-good* (VGEC) if the following two properties hold:

- $(\mathbf{1_V})$ $\forall\, s \in C$, $s \models_{\mathcal{M}} \mathtt{S}(p_1)$;
- $(\mathbf{2_V})$ $\forall\, s \in C$, $s \models_{\mathcal{M}_{\restriction C}} \mathtt{AS}(p_1) \wedge \mathtt{AS}(p_2)$, or equivalently, $s \models_{\mathcal{M}_{\restriction C}} \mathtt{AS}(p_1 \cap p_2)$.

We introduce the following notations: $\mathrm{VGEC}(\mathcal{M})$ is the set of all VGECs of $\mathcal{M}$, and $\mathcal{V} = \cup_{V \in \mathrm{VGEC}(\mathcal{M})} V$ is the set of states that belong to a VGEC in $\mathcal{M}$.

Note that in condition $(\mathbf{1_V})$, $\mathcal{P}_1$ is allowed to leave C to ensure $\mathtt{S}(p_1)$: this is in contrast to condition $(\mathbf{1_U})$ for UGECs, in Definition 5. On the contrary, condition $(\mathbf{2_V})$ is exactly the same as $(\mathbf{2_U})$. From these definitions, it is trivial to see that any UGEC is also a VGEC,

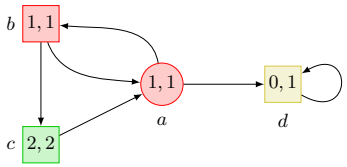


Figure 3 The EC $\{a, b, c\}$ is very-good but not ultra-good, as $\mathcal{P}_1$ has to leave it to ensure $\mathtt{S}(p_1)$.

but the converse is false. Consider Figure 3: $\{a, b, c\}$ is a VGEC. The strategy ensuring $(\mathbf{2_V})$ from a is to go to b, and the strategy ensuring $(\mathbf{1_V})$ from a is to go to d. As we will prove in Lemma 18 and as in all VGECs, $\mathcal{P}_1$ can ensure $a \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>1-\varepsilon}(p_2)$ for any $\varepsilon > 0$. Still, $\{a, b, c\}$ is not a UGEC: no strategy ensures $\mathtt{S}(p_1)$ on $\mathcal{M}_{\downarrow\{a,b,c\}}$, as $\mathcal{P}_2$ can enforce the play $(ab)^\omega$ that has odd maximal priority. This illustrates why the notion of UGEC is too strong when reasoning about threshold probability, hence why we need to introduce VGECs.

Available strategies in VGECs. As for UGECs, we will use witness strategies for $(\mathbf{1_V})$ and $(\mathbf{2_V})$. Deciding if $(\mathbf{1_V})$ holds is solving a classical parity game, in $\mathsf{NP} \cap \mathsf{coNP}$ [19]. Uniform pure memoryless witness strategies exist: let λ_1 be such a witness. For simplicity of presentation, *we assume in the following that all states of* $\mathcal{M}$ *satisfy* $(\mathbf{1_V})$, as otherwise they will trivially not satisfy the properties we consider (as $\mathtt{S}(p_1)$ will not be ensured). For $(\mathbf{2_V})$, we established in Lemma 7 that deciding if it holds is in polynomial time and that uniform randomized memoryless witness strategies exist: let $\lambda_{2,C}$ be one of them.

Reaching VGECs. We prove a strong relationship between the measure of paths that satisfy p_1 and p_2, and the measure of paths that reach VGECs, under any strategy.

▶ **Lemma 17.** *For all $s \in S$, and all $\lambda \in \Lambda$, the following holds:* $\mathbb{P}^{\lambda}_{\mathcal{M},s}[\Diamond \mathcal{V}] \geq \mathbb{P}^{\lambda}_{\mathcal{M},s}[p_1 \cap p_2]$.

Limit-sure satisfaction in VGECs. For each state in a VGEC, we claim that the parity objective p_2 can be satisfied with probability arbitrarily close to one, while ensuring p_1 surely.

▶ **Lemma 18.** *Let $C \in \mathrm{VGEC}(\mathcal{M})$. For all $s \in C$ and $\varepsilon \in (0, 1]$, the following property holds:* $s \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>1-\varepsilon}(p_2)$.

Proof Sketch. As for UGECs, we build a witness strategy based on two simpler ones: λ_1 and $\lambda_{2,C}$. However, our strategy here depends on ε. The rough idea is as follows: the witness strategy λ_ε will play $\lambda_{2,C}$ for longer and longer rounds, and switch to λ_1 forever if $D^{\max}_{\mathrm{even}}(p_1)$ is not visited along one of those rounds. Using Lemma 7 cleverly, we can define the sequence of round lengths in such a way that the probability to play as $\lambda_{2,C}$ forever exceeds $1 - \varepsilon$, yielding the result. ◀

The strict threshold case. We reduce the problem to a reachability problem toward $\mathcal{V}$. The first lemma gives a sufficient condition under which the property is satisfied. Its proof tells us how to construct witness strategies.

▶ **Lemma 19.** *The following holds:* $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>c}(\Diamond \mathcal{V}) \implies s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>c}(p_2)$.

Proof Sketch. We build a witness strategy λ based on (i) $\lambda_{\Diamond \mathcal{V}}$, a strategy that ensures to reach $\mathcal{V}$ with probability $q > c$ from s_0, (ii) λ_1, and (iii) $\lambda_{\varepsilon,C}$ from Lemma 18 for a well-chosen $\varepsilon > 0$ and every VGEC C. The idea is to first play $\lambda_{\Diamond \mathcal{V}}$ long enough so that a VGEC C is reached with probability close to q and, if such a C is reached, to switch to $\lambda_{\varepsilon,C}$ for ε

sufficiently small so that the total probability to satisfy p_2 is higher than c. If no VGEC is reached, λ switches to λ_1, hence ensuring $\mathtt{S}(p_1)$. ◀

This second lemma gives a necessary condition. Its proof uses Lemma 17.

▶ **Lemma 20.** *The following holds:* $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>c}(p_2) \implies s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>c}(\Diamond \mathcal{V})$.

The non-strict threshold case. As we solved the strict case, the only interesting remaining case is when $\mathcal{P}_1$, while surely forcing p_1, can force p_2 with probability c, but no more. The main tool here is UGECs. The first lemma gives a sufficient condition. Its proof is constructive. Recall that $\mathcal{U} = \cup_{U \in \mathrm{UGEC}(\mathcal{M})} U$.

▶ **Lemma 21.** *The following holds:* $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\geq c}(\Diamond \mathcal{U}) \implies s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\geq c}(p_2)$.

Proof Sketch. We define a witness strategy based on (i) $\lambda_{\Diamond \mathcal{U}}$, a witness for $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\geq c}(\Diamond \mathcal{U})$, and (ii) strategies λ_C for every UGEC C: it suffices to play $\lambda_{\Diamond \mathcal{U}}$ as long as no UGEC C is reached and to switch to λ_C when reached, if ever. ◀

The next lemma gives a necessary condition, keeping in mind that we consider the case where $\mathcal{P}_1$ cannot ensure probability strictly larger than c.

▶ **Lemma 22.** *The following holds:* $(s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\geq c}(p_2)) \wedge (s_0 \not\models \mathtt{S}(p_1) \wedge \mathtt{P}_{>c}(p_2)) \implies s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\geq c}(\Diamond \mathcal{U})$.

Algorithm. Based on the reductions shown above, we can now establish an algorithm and complexity results for the threshold problem.

▶ **Theorem 23.** *Given an MDP* $\mathcal{M} = (G = (S, E), S_1, S_2, \delta)$*, a state* $s_0 \in S$*, and two priority functions* $p_i \colon S \to \{1, \dots, d\}$*,* $i \in \{1, 2\}$*, it can be decided in* $\mathsf{NP} \cap \mathsf{coNP}$ *if* $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\sim c}(p_2)$ *for* $\sim \in \{>, \geq\}$ *and* $c \in \mathbb{Q} \cap [0, 1)$*. If the answer is* Yes*, then there exists an infinite-memory witness strategy, and infinite memory is in general necessary. This decision problem is at least as hard as solving parity games.*

Proof. The algorithm can be sketched as follows:

1. Remove from $\mathcal{M}$ all states where $\mathtt{S}(p_1)$ does not hold, as well as their attractor for $\mathcal{P}_2$: if s_0 is removed, then answer No. Let $\mathcal{M}'$ be the remaining MDP. This operation is in $\mathsf{NP} \cap \mathsf{coNP}$ as it consists in solving a classical parity game [19].
2. Compute the set $\mathcal{V}$ representing the union of VGECs in $\mathcal{M}'$. This can be done in polynomial time by computing the maximal ECs of $\mathcal{M}'$ and applying Lemma 7 to check condition $(\mathbf{2_V})$ for each of them (condition $(\mathbf{1_V})$ holds thanks to the previous step).
3. Decide if $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>c}(\Diamond \mathcal{V})$ using Theorem 4. This is in $\mathsf{NP} \cap \mathsf{coNP}$. If it holds, then answer Yes. If it does not hold and $\sim$ is $>$, then answer No, otherwise, i.e., if $\sim$ is $\geq$, continue with the next step.
4. Use the sub-algorithm described in Theorem 15 to compute the set $\mathcal{U}$ representing the union of UGECs in $\mathcal{M}'$. This is in $\mathsf{NP} \cap \mathsf{coNP}$.
5. Decide if $s_0 \models \mathtt{S}(p_1) \wedge \mathtt{P}_{\geq c}(\Diamond \mathcal{U})$ using Theorem 4. This is in $\mathsf{NP} \cap \mathsf{coNP}$. If it holds, answer Yes, otherwise answer No.

The correctness of this algorithm follows from Lemma 19, Lemma 20, Lemma 21, and Lemma 22. It belongs to $\mathsf{P}^{\mathsf{NP} \cap \mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$ [6], and it trivially generalizes classical parity games (e.g., by taking $p_2 \colon s \mapsto 0$ for all $s \in S$).

Finally, let us discuss strategies. Witness strategies for the case $>$ (resp. $\geq$) were described in Lemma 19 (resp. Lemma 21). In both cases, infinite memory is in general required, because it is in general necessary to play optimally in both VGECs and UGECs. For UGECs, see Theorem 15 for an example. For VGECs, consider the VGEC $\{a, b, c\}$ in the MDP of Figure 3. We claim that for every finite-memory strategy λ ensuring $\mathtt{S}(p_1)$, the probability to ensure p_2 is zero, hence there is no finite-memory witness for $a \models \mathtt{S}(p_1) \wedge \mathtt{P}_{>1-\varepsilon}(p_2)$. As argued for the UGEC case, in order to ensure p_1 on the play in which $\mathcal{P}_2$ always goes to a from b, $\mathcal{P}_1$ must go to d at some point, and because of the finite memory of λ, he must do it after a bounded number of steps along which c is not visited: say n steps. Again, the probability to do it will be bounded from below by a strictly positive constant, here $2^{-\frac{n}{2}}$ (the probability that $\mathcal{P}_2$ chooses a for $\frac{n}{2}$ times in a row), all along a consistent play. Therefore, $\mathcal{P}_1$ will almost-surely go to d, and p_2 will actually be satisfied with probability zero. ◀

6 Conclusion

We further extended the beyond worst-case synthesis framework by studying the case of two parity objectives and proved $\mathsf{NP} \cap \mathsf{coNP}$ membership for all considered variants.

Our algorithms can easily be generalized to more than two parity objectives as long as we consider only the $\mathtt{S}$ and $\mathtt{AS}$ operators. Indeed, we have that for any MDP $\mathcal{M}$, any state s in $\mathcal{M}$, and any number of priority functions $p_1, \dots p_n$, it holds that $s \models \bigwedge_i \mathtt{S}(p_i) \bigwedge_j \mathtt{AS}(p_j) \iff s \models \mathtt{S}\big(\bigwedge_i p_i\big) \wedge \mathtt{AS}\big(\bigwedge_j p_j\big)$, and it is easy to reduce the latter problem to $s' \models \mathtt{S}(p') \wedge \mathtt{AS}(p'')$ on a (larger) MDP $\mathcal{M}'$, using classical techniques (e.g., any conjunction of parity objectives can be expressed as a Muller condition [13], that in turn can be transformed into a single parity condition on a larger graph [20]). Extending this generalization to the operator $\mathtt{P}_{\sim c}$ is more challenging and would require to mix our techniques to methods for percentile queries [24]: an interesting direction for future work.

Another question is the limits of finite-memory strategies. We saw that in general, infinite memory is needed. We would like to investigate under which additional conditions finite-memory strategies suffice, and to develop corresponding algorithms.

References

1. Shaull Almagor, Orna Kupferman, and Yaron Velner. Minimizing expected cost under hard boolean constraints, with applications to quantitative synthesis. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPIcs*, pages 9:1–9:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.9.
2. Shaull Almagor, Orna Kupferman, and Yaron Velner. Minimizing expected cost under hard boolean constraints, with applications to quantitative synthesis. *CoRR*, abs/1604.07064, 2016. URL: http://arxiv.org/abs/1604.07064.
3. Christel Baier, Marcus Größer, and Frank Ciesinski. Quantitative analysis under fairness constraints. In Zhiming Liu and Anders P. Ravn, editors, *Automated Technology for Verification and Analysis, 7th International Symposium, ATVA 2009, Macao, China, October 14-16, 2009. Proceedings*, volume 5799 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2009. doi:10.1007/978-3-642-04761-9_12.
4. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.
5. Raphaël Berthon, Mickael Randour, and Jean-François Raskin. Threshold constraints with guarantees for parity objectives in Markov decision processes. *CoRR*, abs/1702.05472, 2017. URL: http://arxiv.org/abs/1702.05472.

6 Gilles Brassard. A note on the complexity of cryptography (corresp.). *IEEE Transactions on Information Theory*, 25(2):232–233, 1979.

7 Tomás Brázdil, Antonín Kucera, and Petr Novotný. Optimizing the expected mean payoff in energy Markov decision processes. In Cyrille Artho, Axel Legay, and Doron Peled, editors, *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*, volume 9938 of *Lecture Notes in Computer Science*, pages 32–49, 2016. doi:10.1007/978-3-319-46520-3_3.

8 Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A. Pérez, Mickael Randour, Jean-François Raskin, Ocan Sankur, and Mathieu Sassolas. Non-zero sum games for reactive synthesis. In Adrian-Horia Dediu, Jan Janousek, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, volume 9618 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2016. doi:10.1007/978-3-319-30000-9_1.

9 Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Expectations or guarantees? I want it all! A crossroad between games and MDPs. In Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi, editors, *Proceedings 2nd International Workshop on Strategic Reasoning, SR 2014, Grenoble, France, April 5-6, 2014*, volume 146 of *EPTCS*, pages 1–8, 2014. doi:10.4204/EPTCS.146.1.

10 Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science, STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 199–213. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.199.

11 Véronique Bruyère, Quentin Hautem, and Mickael Randour. Window parity games: an alternative approach toward parity games with time bounds. In Domenico Cantone and Giorgio Delzanno, editors, *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016*, volume 226 of *EPTCS*, pages 135–148, 2016. doi:10.4204/EPTCS.226.10.

12 Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Mean-payoff parity games. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 178–187. IEEE Computer Society, 2005. doi:10.1109/LICS.2005.26.

13 Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized parity games. In Helmut Seidl, editor, *Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 24-April 1, 2007, Proceedings*, volume 4423 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2007. doi:10.1007/978-3-540-71389-0_12.

14 Lorenzo Clemente and Jean-François Raskin. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 257–268. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.33.

15 Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Axel Legay, Didier Lime, Mathias Grund Sørensen, and Jakob Haahr Taankvist. On time with minimal expected cost! In Franck Cassez and Jean-François Raskin, editors, *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings*, volume 8837 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2014. doi:10.1007/978-3-319-11936-6_10.

16 E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ-calculus. In Costas Courcoubetis, editor, *Computer Aided Verification, 5th International Conference, CAV'93, Elounda, Greece, June 28 - July 1, 1993, Proceedings*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 1993. doi:10.1007/3-540-56922-7_32.

17 Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer, 1997.

18 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

19 Marcin Jurdzinski. Deciding the winner in parity games is in UP ∩ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998. doi:10.1016/S0020-0190(98)00150-1.

20 Christof Löding. Optimal bounds for transformations of omega-automata. In C. Pandu Rangan, Venkatesh Raman, and Ramaswamy Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science, 19th Conference, Chennai, India, December 13-15, 1999, Proceedings*, volume 1738 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1999. doi:10.1007/3-540-46691-6_8.

21 Mickael Randour. Automated synthesis of reliable and efficient systems through game theory: A case study. In *Proc. of ECCS 2012*, Springer Proceedings in Complexity XVII, pages 731–738. Springer, 2013. doi:10.1007/978-3-319-00395-5_90.

22 Mickael Randour. Reconciling rationality and stochasticity: Rich behavioral models in two-player games. *CoRR*, abs/1603.05072, 2016. *GAMES 2016, the 5th World Congress of the Game Theory Society, Maastricht, Netherlands*. URL: http://arxiv.org/abs/1603.05072.

23 Mickael Randour, Jean-François Raskin, and Ocan Sankur. Variations on the stochastic shortest path problem. In Deepak D'Souza, Akash Lal, and Kim Guldstrand Larsen, editors, *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*, volume 8931 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2015. doi:10.1007/978-3-662-46081-8_1.

24 Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. *Formal Methods in System Design*, 50(2):207–248, 2017. doi:10.1007/s10703-016-0262-7.

Synchronizability of Communicating Finite State Machines is not Decidable*

Alain Finkel[1] and Etienne Lozes[2]

1 **LSV, ENS Cachan, CNRS, Cachan, France**
`finkel@lsv.fr`
2 **LSV, ENS Cachan, CNRS, Cachan, France**
`lozes@lsv.fr`

Abstract

A system of communicating finite state machines is *synchronizable* [1, 4] if its send trace semantics, i.e. the set of sequences of sendings it can perform, is the same when its communications are FIFO asynchronous and when they are just rendez-vous synchronizations. This property was claimed to be decidable in several conference and journal papers [1, 4, 3, 2] for either mailboxes ($*$-1) or peer-to-peer (1-1) communications, thanks to a form of small model property. In this paper, we show that this small model property does not hold neither for mailbox communications, nor for peer-to-peer communications, therefore the decidability of synchronizability becomes an open question. We close this question for peer-to-peer communications, and we show that synchronizability is actually undecidable. We show that synchronizability is decidable if the topology of communications is an oriented ring. We also show that, in this case, synchronizability implies the absence of unspecified receptions and orphan messages, and the channel-recognizability of the reachability set.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases verification, distributed system, asynchronous communications, choreographies

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.122

1 Introduction

Asynchronous distributed systems are error prone not only because they are difficult to program, but also because they are difficult to execute in a reproducible way. The slack of communications, measured by the number of messages that can be buffered in a same communication channel, is not always under the control of the programmer, and even when it is, it may be delicate to choose the right size of the communication buffers.

Slack elasticity of a distributed system with asynchronous communications is the property that the "observable behaviour" of the system is the same whatever the slack of communications is. There are actually as many notions of slack elasticity as there are notions of observable behaviours (and of distributed systems). Slack elasticity has been studied in various contexts: for hardware design [16], with the goal of ensuring that some code transformations are semantic-preserving, for parallel programming in MPI [18, 19], for ensuring the absence of deadlocks and other bugs, or more recently for web services and choreographies [1, 4, 2], for verifying various properties, among which choreography realizability [3].

* This is a proceedings version. The full version is [11], `https://arxiv.org/abs/1702.07213`.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 122; pp. 122:1–122:14


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

This paper focuses on *synchronizability* [1], a special form of slack elasticity that was defined by Basu and Bultan for analyzing choreographies. Synchronizability is the slack elasticity of the send trace semantics of the system: a system of communicating finite state machines is synchronizable if any asynchronous trace can be mimicked by a synchronous one that contains the same send actions in the same order. Synchronizability was claimed decidable first for mailbox communications [4], where each peer stores all incoming messages in a unique mailbox in a FIFO fashion. Later, the decidability claim was extended to peer-to-peer communications [2], where there is a FIFO queue for every pair of peers. Synchronizability seemed to contrast with many other properties of systems of communicating finite state machines (including deadlock-freedom, absence of orphan messages, boundedness, etc) that are undecidable for systems of just two machines [6]. The proof relied on the claim that synchronizability would be the same as 1-synchronizability, which states that any 1-bounded trace can be mimicked by a synchronous trace.

In this paper, we show that the two claims are actually false: 1-synchronizability does not imply synchronizability, and at least for peer-to-peer communications, synchronizability is undecidable. We also show that the two claims hold, however, if we restrict to systems where the communication topology is an oriented, unidirectional ring, in particular the topology of a system with two peers only. While proving that 1-synchronizability implies synchronizability for ring topologies we also show that 1-synchronizability implies the absence of unspecified receptions and orphan messages, and that the reachability set is channel-recognizable.

Outline. The paper focuses on the peer-to-peer communication model. Section 2 introduces all notions of communicating finite state machines and synchronizability. In Section 3, we show that synchronizability is undecidable. Section 4 shows the decidability of synchronizability on ring topologies. Section 5 concludes with discussions and open problems about other communication models, in particular the mailbox communication model that was the first and the most studied model in previous works on synchronizability. Due to space constraints, several proofs are omitted and can be found in a companion long version [11].

Related Work. The analysis of systems of communicating finite state machines has always been a very active topic of research. Systems with channel-recognizable (aka QDD [5] representable) reachability sets are known to enjoy a decidable reachability problem [17]. Heussner *et al* developed a CEGAR approach based on regular model-checking [13]. Classifications of communication topologies according to the decidability of the reachability problems are known for FIFO, FIFO+lossy, and FIFO+bag communications [8, 9]. In [15, 14], the bounded context-switch reachability problem for communicating machines extended with local stacks modeling recursive function calls is shown decidable under various assumptions. Session types dialects have been introduced for systems of communicating finite state machines [10], and were shown to enforce various desirable properties. Existentially-bounded systems are systems of communicating finite state machines that were studied in a language-theoretic perspective: in [12], in particular, correspondences have been established among message sequence charts languages defined on the one hand by (universally/existentially bounded) systems of communicating machines and on the other hand by monadic second order logic over partial orders and automata Whether a system of communicating machines is existentially bounded, respectively existentially k-bounded for a fixed k, is undecidable in the general case, but it is unknown whether it remains undecidable for systems that are non-blocking.

2 Preliminaries

Messages and topologies. A *message set* M is a tuple $\langle \Sigma_M, p, \mathsf{src}, \mathsf{dst}\rangle$ where Σ_M is a finite set of letters (more often called messages), $p \geq 1$ and $\mathsf{src}, \mathsf{dst}$ are functions that associate to every letter $a \in \Sigma$ naturals $\mathsf{src}(a) \neq \mathsf{dst}(a) \in \{1, \ldots, p\}$. We often write $a^{i\to j}$ for a message a such that $\mathsf{src}(a) = i$ and $\mathsf{dst}(a) = j$; we often identify M and Σ_M and write for instance $M = \{a_1^{i_1\to j_1}, a_2^{i_2\to j_2}, \ldots\}$ instead of $\Sigma_M = \ldots$, or $w \in M^*$ instead of $w \in \Sigma_M^*$. The communication topology associated to M is the graph G_M with vertices $\{1, \ldots, p\}$ and with an edge from i to j if there is a message $a \in \Sigma_M$ such that $\mathsf{src}(a) = i$ and $\mathsf{dst}(a) = j$. G_M is an *oriented ring* if the set of edges of G_M is $\{(i, j) \mid i + 1 = j \bmod p\}$.

Traces. An *action* λ over M is either a send action $!a$ or a receive action $?a$, with $a \in \Sigma_M$. The peer $\mathsf{peer}(\lambda)$ of action λ is defined as $\mathsf{peer}(!a) = \mathsf{src}(a)$ and $\mathsf{peer}(?a) = \mathsf{dst}(a)$. We write $\mathsf{Act}_{i,M}$ for the set of actions of peer i and Act_M for the set of all actions over M. A *M-trace* τ is a finite (possibly empty) sequence of actions. We write Act_M^* for the set of M-traces, ϵ for the empty M-trace, and $\tau_1 \cdot \tau_2$ for the concatenation of two M-traces. We sometimes write $!?a$ for $!a \cdot ?a$. A M-trace τ is a prefix of υ, $\tau \leq_{\mathsf{pref}} \upsilon$ if there is θ such that $\upsilon = \tau \cdot \theta$. The prefix closure $\downarrow S$ of a set of M-traces S is the set $\{\tau \in \mathsf{Act}_M^* \mid \text{there is } \upsilon \in S \text{ such that } \tau \leq_{\mathsf{pref}} \upsilon\}$. For a M-trace τ and peer ids $i, j \in \{1, \ldots, p\}$ we write

- $\mathsf{send}(\tau)$ (resp. $\mathsf{recv}(\tau)$) for the sequence of messages sent (resp. received) during τ, *i.e.* $\mathsf{send}(!a) = a$, $\mathsf{send}(?a) = \epsilon$, and $\mathsf{send}(\tau_1 \cdot \tau_2) = \mathsf{send}(\tau_1) \cdot \mathsf{send}(\tau_2)$ (resp. $\mathsf{recv}(!a) = \epsilon$, $\mathsf{recv}(?a) = a$, and $\mathsf{recv}(\tau_1 \cdot \tau_2) = \mathsf{recv}(\tau_1) \cdot \mathsf{recv}(\tau_2)$).
- $\mathsf{onPeer}_i(\tau)$ for the M-trace of actions λ in τ such that $\mathsf{peer}(\lambda) = i$.
- $\mathsf{onChannel}_{i\to j}(\tau)$ for the M-trace of actions λ in τ such that $\lambda \in \{!a, ?a\}$ for some $a \in M$ with $\mathsf{src}(a) = i$ and $\mathsf{dst}(a) = j$.
- $\mathsf{buffer}_{i\to j}(\tau)$ for the word $w \in M^*$, if it exists, such that $\mathsf{send}(\mathsf{onChannel}_{i\to j}(\tau)) = \mathsf{recv}(\mathsf{onChannel}_{i\to j}(\tau)) \cdot w$.

A M-trace τ is *FIFO* (resp. a *k-bounded FIFO*, for $k \geq 1$) if for all $i, j \in \{1, \ldots, p\}$, for all prefixes τ' of τ, $\mathsf{buffer}_{i\to j}(\tau')$ is defined (resp. defined and of length at most k). A M-trace is *synchronous* if it is of the form $!?a_1 \cdot !?a_2 \cdots !?a_k$ for some $k \geq 0$ and $a_1, \ldots, a_k \in M$. In particular, a synchronous M-trace is a 1-bounded FIFO M-trace (but the converse is false). A M-trace τ is *stable* if $\mathsf{buffer}_{i\to j}(\tau) = \epsilon$ for all $i \neq j \in \{1, \ldots, p\}$.

Two M-traces τ, υ are *causal-equivalent* $\tau \overset{\text{causal}}{\sim} \upsilon$ if

1. τ, υ are FIFO, and
2. for all $i \in \{1, \ldots, p\}$, $\mathsf{onPeer}_i(\tau) = \mathsf{onPeer}_i(\upsilon)$.

The relation $\overset{\text{causal}}{\sim}$ is a congruence with respect to concatenation. Intuitively, $\tau \overset{\text{causal}}{\sim} \upsilon$ if τ is obtained from υ by iteratively commuting adjacent actions that are not from the same peer and do not form a "matching send/receive pair".

Peers, systems, configurations. A system (of communicating machines) over a message set M is a tuple $\mathcal{S} = \langle \mathcal{P}_1, \ldots, \mathcal{P}_p\rangle$ where for all $i \in \{1, \ldots, p\}$, the peer $\mathcal{P}_i$ is a finite state automaton $\langle Q_i, q_{0,i}, \Delta_i\rangle$ over the alphabet $\mathsf{Act}_{i,M}$ and with (implicitly) Q_i as the set of accepting states. We write $L(\mathcal{P}_i)$ for the set of M-traces that label a path in $\mathcal{P}_i$ starting at the initial state $q_{0,i}$.

Let the system $\mathcal{S}$ be fixed. A *configuration* γ of $\mathcal{S}$ is a tuple $(q_1, \ldots, q_p, w_{1,2}, \ldots, w_{p-1,p})$ where q_i is a state of $\mathcal{P}_i$ and for all $i \neq j$, $w_{i,j} \in M^*$ is the content of channel $i \to j$. A configuration is *stable* if $w_{i,j} = \epsilon$ for all $i, j \in \{1, \ldots, p\}$ with $i \neq j$.

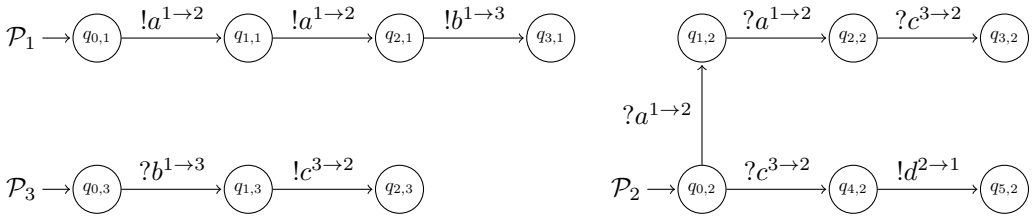


Figure 1 System of Example 1 and Theorem 3.

Let $\gamma = (q_1, \dots, q_p, w_{1,2}, \dots, w_{p-1,p})$, $\gamma' = (q'_1, \dots, q'_p, w'_{1,2}, \dots, w'_{p-1,p})$ and $m \in M$ with $\mathsf{src}(m) = i$ and $\mathsf{dst}(m) = j$. We write $\gamma \xrightarrow{!m}_{\mathcal{S}} \gamma'$ (resp. $\gamma \xrightarrow{?m}_{\mathcal{S}} \gamma'$) if $(q_i, !m, q'_i) \in \Delta_i$ (resp. $(q_j, ?m, q'_j) \in \Delta_j$), $w'_{i,j} = w_{i,j} \cdot m$ (resp. $w_{i,j} = m \cdot w'_{i,j}$) and for all k, ℓ with $k \neq i$ (resp. with $k \neq j$), $q_k = q'_k$ and $w'_{k,\ell} = w_{k,\ell}$ (resp. $w'_{\ell,k} = w_{\ell,k}$). If $\tau = \lambda_1 \cdot \lambda_2 \cdots \lambda_n$, we write $\xrightarrow{\tau}_{\mathcal{S}}$ for $\xrightarrow{\lambda_1}_{\mathcal{S}} \xrightarrow{\lambda_2}_{\mathcal{S}} \dots \xrightarrow{\lambda_n}_{\mathcal{S}}$. We often write $\xrightarrow{\tau}$ instead of $\xrightarrow{\tau}_{\mathcal{S}}$ when $\mathcal{S}$ is clear from the context. The *initial configuration* of $\mathcal{S}$ is the stable configuration $\gamma_0 = (q_{0,1}, \dots, q_{0,p}, \epsilon, \dots, \epsilon)$. A M-trace τ is a trace of system $\mathcal{S}$ if there is γ such that $\gamma_0 \xrightarrow{\tau} \gamma$. Equivalently, τ is a trace of $\mathcal{S}$ if

1. it is a FIFO trace, and
2. for all $i \in \{1, \dots, p\}$, $\mathsf{onPeer}_i(\tau) \in L(\mathcal{P}_i)$.

For $k \geq 1$, we write $\mathsf{Traces}_k(\mathcal{S})$ for the set of k-bounded traces of $\mathcal{S}$, $\mathsf{Traces}_0(\mathcal{S})$ for the set of synchronous traces of $\mathcal{S}$, and $\mathsf{Traces}_\omega(\mathcal{S})$ for $\bigcup_{k\geq 0} \mathsf{Traces}_k(\mathcal{S})$.

▶ **Example 1.** Consider the message set $M = \{a^{1\to2}, b^{1\to3}, c^{3\to2}, d^{2\to1}\}$ and the system $\mathcal{S} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ where $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ are as depicted in Fig. 1. Then

$$\begin{aligned} L(\mathcal{P}_1) &= \downarrow\{!a^{1\to2} \cdot !a^{1\to2} \cdot !b^{1\to3}\} \\ L(\mathcal{P}_2) &= \downarrow\{?a^{1\to2} \cdot ?a^{1\to2} \cdot ?c^{3\to2} \ , \ ?c^{3\to2} \cdot !d^{2\to1}\} \\ L(\mathcal{P}_3) &= \downarrow\{?b^{1\to3} \cdot !c^{3\to2}\}. \end{aligned}$$

An example of a stable trace is $!a^{1\to2} \cdot !a^{1\to2} \cdot !?b^{1\to3} \cdot !c^{3\to2} \cdot ?a^{1\to2} \cdot ?a^{1\to2} \cdot ?c^{3\to2}$. Let $\tau = !a^{1\to2} \cdot !a^{1\to2} \cdot !?b^{1\to3} \cdot !?c^{3\to2} \cdot !d^{2\to1}$. Then $\tau \in \mathsf{Traces}_2(\mathcal{S})$ is a 2-bounded trace of the system $\mathcal{S}$, and $\gamma_0 \xrightarrow{\tau} (q_{3,1}, q_{5,2}, q_{2,3}, a^{1\to2}a^{1\to2}, \epsilon, d^{2\to1}, \epsilon, \epsilon, \epsilon)$.

Two traces τ_1, τ_2 are $\mathcal{S}$-*equivalent*, $\tau_1 \overset{\mathcal{S}}{\sim} \tau_2$, if $\tau_1, \tau_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ and there is γ such that $\gamma_0 \xrightarrow{\tau_i} \gamma$ for both $i = 1, 2$. It follows from the definition of $\overset{\text{causal}}{\sim}$ that if $\tau_1 \overset{\text{causal}}{\sim} \tau_2$ and $\tau_1, \tau_2 \in \mathsf{Traces}_\omega(\mathcal{S})$, then $\tau_1 \overset{\mathcal{S}}{\sim} \tau_2$.

Synchronizability. Following [4], we define the observable behaviour of a system as its set of send traces enriched with their final configurations when they are stable. Formally, for any $k \geq 0$, we write $\mathcal{J}_k(\mathcal{S})$ and $\mathcal{I}_k(\mathcal{S})$ for the sets

$$\begin{aligned} \mathcal{J}_k(\mathcal{S}) &= \{\mathsf{send}(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{S})\} \\ \mathcal{I}_k(\mathcal{S}) &= \mathcal{J}_k(\mathcal{S}) \cup \{(\mathsf{send}(\tau), \gamma) \mid \gamma_0 \xrightarrow{\tau} \gamma, \gamma \text{ stable}, \tau \in \mathsf{Traces}_k(\mathcal{S})\}. \end{aligned}$$

Synchronizability is then defined as the slack elasticity of these observable behaviours.

▶ **Definition 2** (Synchronizability [1, 4])**.** A system $\mathcal{S}$ is *synchronizable* if $\mathcal{I}_0(\mathcal{S}) = \mathcal{I}_\omega(\mathcal{S})$. $\mathcal{S}$ is called *language synchronizable* if $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_\omega(\mathcal{S})$.

For convenience, we also introduce a notion of k-synchronizability: for $k \geq 1$, a system $\mathcal{S}$ is k-*synchronizable* if $\mathcal{I}_0(\mathcal{S}) = \mathcal{I}_k(\mathcal{S})$, and *language* k-synchronizable if $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_k(\mathcal{S})$. A system is therefore (language) synchronizable if and only if it is (language) k-synchronizable for all $k \geq 1$.

▶ **Theorem 3.** *There is a system $\mathcal{S}$ that is 1-synchronizable, but not synchronizable.*

Proof. Consider again the system $\mathcal{S}$ of Example 1. Let $\gamma_{ijk} := (q_{i,1}, q_{j,2}, q_{k,3}, \epsilon, \ldots, \epsilon)$. Then

$$\begin{array}{lcl} \mathcal{J}_0(\mathcal{S}) & = & \downarrow \{a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3} \cdot c^{3\to 2}\} \\ \mathcal{J}_1(\mathcal{S}) & = & \mathcal{J}_0(\mathcal{S}) \\ \mathcal{J}_2(\mathcal{S}) & = & \downarrow \{a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3} \cdot c^{3\to 2} \cdot d^{2\to 1}\} \\ \mathcal{I}_k(\mathcal{S}) & = & \mathcal{J}_k(\mathcal{S}) \cup \mathsf{Stab} \qquad \text{for all } k \geq 0 \end{array}$$

where $\mathsf{Stab} = \{(\epsilon, \gamma_0), (a^{1\to 2}, \gamma_{101}), (a^{1\to 2} \cdot a^{1\to 2}, \gamma_{202}), (a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3}, \gamma_{312}), (a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3} \cdot c^{3\to 2}, \gamma_{323})\}$. ◀

This example contradicts Theorem 4 in [2], which stated that $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_1(\mathcal{S})$ implies $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_\omega(\mathcal{S})$. This also shows that the decidability of synchronizability for peer-to-peer communications is open despite the claim in [2]. The next section closes this question.

▶ Remark. In Section 5, we give a counter-example that addresses communications with mailboxes, *i.e.* the first communication model considered in all works about synchronizability, and we list several other published theorems that our counter-example contradicts.

3 Undecidability of Synchronizability

In this section, we show the undecidability of synchronizability for systems with at least three peers. The key idea is to reduce a decision problem on a FIFO automaton $\mathcal{A}$, i.e. an automaton that can both enqueue and dequeue messages in a unique channel, to the synchronizability of a system $\mathcal{S}_\mathcal{A}$. The reduction is quite delicate, because synchronizability constrains a lot the way $\mathcal{S}_\mathcal{A}$ can be defined (a hint for that being that $\mathcal{S}_\mathcal{A}$ *must* involve three peers). It is also delicate to reduce from a classical decision problem on FIFO automata like *e.g.* the reachability of a control state, and we first establish the undecidability of a well-suited decision problem on FIFO automata, roughly the reception of a message m with some extra constraints. We can then construct a system $\mathcal{S}''_{\mathcal{A},m}$ such that the synchronizability of $\mathcal{S}''_{\mathcal{A},m}$ is equivalent to the non-reception of the special message m in $\mathcal{A}$.

A *FIFO automaton* is a finite state automaton $\mathcal{A} = \langle Q, \mathsf{Act}_\Sigma, \Delta, q_0 \rangle$ over an alphabet of the form Act_Σ for some finite set of letters Σ with all states being accepting states. A FIFO automaton can be thought as a system with only one peer, with the difference that, according to our definition of systems, a peer can only send messages to peers different from itself, whereas a FIFO automaton enqueues and dequeues letters in a unique FIFO queue, and thus, in a sense, "communicates with itself". All notions we introduced for systems are obviously extended to FIFO automata. In particular, a configuration of $\mathcal{A}$ is a tuple $\gamma = (q, w) \in Q \times \Sigma^*$, it is stable if $w = \epsilon$, and the transition relation $\gamma \xrightarrow{\tau} \gamma'$ is defined exactly the same way as for systems. For technical reasons, we consider two mild restrictions on FIFO automata:

(***R1***) for all $\gamma_0 \xrightarrow{\tau} (q, w)$, either $\tau = \epsilon$ or $w \neq \epsilon$ (in other words, all reachable configurations are unstable, except the initial one);

(***R2***) for all $(q_0, \lambda, q) \in \Delta$, $\lambda = !a$ for some $a \in \Sigma$ (in other words, there is no receive action labeling a transition from the initial state).

▶ **Lemma 4.** *The following decision problem is undecidable.*
Input *A FIFO automaton $\mathcal{A}$ that satisfies (R1) and (R2), and a message m.*
Question *Is there a M-trace τ such that $\tau \cdot ?m \in \mathsf{Traces}_\omega(\mathcal{A})$?*

Proof. This kind of result is often considered folklore, but it seems it could be informative to detail a possible construction. We reduce from the existence of a finite tiling given a set of tiles and a pair of initial and final tiles. Intuitively, we construct a FIFO automaton that outputs the first row of the tiling, storing it into the queue, and then for all next row $i+1$, the automaton outputs the row tile after tile, popping a tile of row i in the queue in between so as to check that each tile of row $i+1$ vertically coincides with the corresponding tile of row i. Consider a tuple $\mathcal{T} = \langle T, t_0, t_F, H, V\rangle$ where T is a finite set of tiles $t_0, t_F \in T$ are initial and final tiles, and $H, V \subseteq T \times T$ are horizontal and vertical compatibility relations. Without loss of generality, we assume that there is a "padding tile" $\square$ such that $(t, \square) \in H \cap V$ for all $t \in T$. For a natural $n \geq 1$, a n-tiling is a function $f : \mathbb{N} \times \{1, \ldots, n\} \to T$ such that

1. $f(0,0) = t_0$,
2. there are $(i_F, j_F) \in \mathbb{N} \times \{1, \ldots, n\}$ such that $f(i_F, j_F) = t_F$,
3. $(f(i,j), f(i,j+1)) \in H$ for all $(i,j) \in \mathbb{N} \times \{1, \ldots, n-1\}$, and
4. $(f(i,j), f(i+1,j)) \in V$ for all $(i,j) \in \mathbb{N} \times \{1, \ldots, n\}$.

The problem of deciding, given a tuple $\mathcal{T} = \langle T, t_0, t_F, H, V\rangle$, whether there is some $n \geq 1$ for which there exists a n-tiling, is undecidable.[1] Let $\mathcal{T} = \langle T, t_0, t_F, H, V\rangle$ be fixed. We define the FIFO automaton $\mathcal{A}_\mathcal{T} = \langle Q, \Sigma, \Delta, q_0\rangle$ with $Q = \{q_{t,0}, q_{\downarrow=t}, q_{\leftarrow=t}, q_{\leftarrow=t,\downarrow=t'} \mid t \in T, t' \in T \cup \{\$\}\} \cup \{q_0, q_1\}$, $\Sigma = T \cup \{\$\}$, and $\Delta \subseteq Q \times \mathsf{Act}_\Sigma \times Q$, with

$$\begin{array}{rcl}
\Delta & = & \{(q_0, !t_0, q_{t_0,0})\} \cup \{(q_{t,0}, !t', q_{t',0}) \mid (t,t') \in H\} \cup \{(q_{t,0}, !\$, q_1) \mid t \in T\} \\
& \cup & \{(q_1, ?t, q_{\downarrow=t}) \mid t \in T)\} \cup \{(q_{\downarrow=t}, !t', q_{\leftarrow=t'}) \mid (t,t') \in V\} \\
& \cup & \{(q_{\leftarrow=t}, ?t', q_{\leftarrow=t,\downarrow=t'}) \mid t \in T, t' \in T \cup \{\$\}\} \\
& \cup & \{(q_{\leftarrow=t,\downarrow=t'}, !t'', q_{\leftarrow=t''}) \mid (t,t'') \in H \text{ and } (t',t'') \in V\} \\
& \cup & \{(q_{\leftarrow=t,\downarrow=\$}, !\$, q_1) \mid t \in T\}
\end{array}$$

Therefore, any execution of $\mathcal{A}_\mathcal{T}$ is of the form

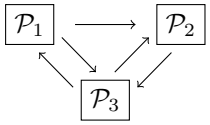
$$!t_{1,1} \cdot !t_{1,2} \cdots !t_{1,n} \cdot !\$ \cdot ?t_{1,1} \cdot !t_{2,1} \cdot ?t_{1,2} \cdot !t_{2,2} \cdots !t_{2,n} \cdot ?\$ \cdot !\$ \cdot ?t_{2,1} \cdot !t_{3,1} \cdots$$

where $t_{1,1} = t_0$, $(t_{i,j}, t_{i+1,j}) \in V$ and $(t_{i,j}, t_{i,j+1}) \in H$. The following two are thus equivalent:

1. there is $n \geq 1$ such that $\mathcal{T}$ admits a n-tiling
2. there is a trace $\tau \in \mathsf{Traces}_\omega(\mathcal{A})$ that contains $?t_F$. ◀

Let us now fix a FIFO automaton $\mathcal{A} = \langle Q_\mathcal{A}, \mathsf{Act}_\Sigma, \Delta_\mathcal{A}, q_0\rangle$ that satisfies $(R1)$ and $(R2)$. Let $M = M_1 \cup M_2 \cup M_3$ be such that all messages of Σ can be exchanged among all peers in all directions but $2 \to 1$, *i.e.*

$$\begin{array}{rcl}
M_1 & = & \{a^{1\to2}, a^{1\to3}, a^{3\to1} \mid a \in \Sigma\} \\
M_2 & = & \{a^{3\to2}, a^{1\to2}, a^{2\to3} \mid a \in \Sigma\} \\
M_3 & = & \{a^{1\to3}, a^{3\to1}, a^{3\to2}, a^{2\to3} \mid a \in \Sigma\}
\end{array}$$



Intuitively, we want $\mathcal{P}_1$ to mimick $\mathcal{A}$'s decisions and the channel $1 \to 2$ to mimick $\mathcal{A}$'s queue as follows. When $\mathcal{A}$ would enqueue a letter a , peer 1 sends $a^{1\to2}$ to peer 2, and when $\mathcal{A}$ would dequeue a letter a, peer 1 sends to peer 2 via peer 3 the order to dequeue a, and waits for the acknowledgement that the order has been correcly executed. Formally, let $\mathcal{P}_1 = \langle Q_1, q_{0,1}, \Delta_1\rangle$ be defined by $Q_1 = Q_\mathcal{A} \uplus \{q_\delta \mid \delta \in \Delta_\mathcal{A}\}$ and $\Delta_1 = \{(q, !a^{1\to2}, q') \mid$

[1] Note that, due to the presence of the padding tile, this problem is equivalent to the problem of the existence of a finite rectangular tiling that contains t_0 at the beginning of the first row and t_F anywhere in the rectangle, which in turn is equivalent to the termination of a Turing machine.

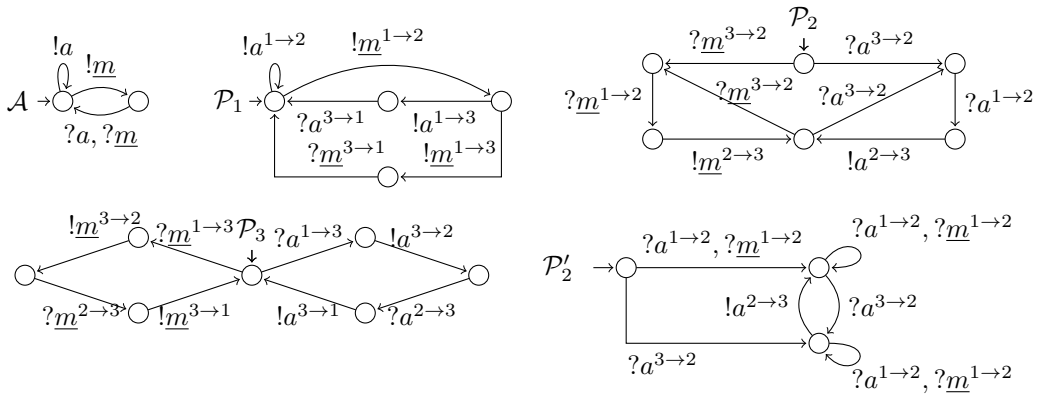


Figure 2 The FIFO automaton $\mathcal{A}$ of Example 5 and its associated systems $\mathcal{S}_\mathcal{A} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ and $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3 \rangle$. The sink state $q_\perp$ and the transitions $q \xrightarrow{?\underline{m}^{3\to2}} q_\perp$ are omitted in the representation of $\mathcal{P}'_2$.

$(q, !a, q') \in \Delta_\mathcal{A}\} \cup \{(q, !a^{1\to3}, q_\delta), (q_\delta, ?a^{3\to1}, q') \mid \delta = (q, ?a, q') \in \Delta_\mathcal{A}\}$. The roles of peers 2 and 3 is then rather simple: peer 3 propagates all messages it receives, and peer 2 executes all orders it receives and sends back an acknowledgement when this is done. Let $\mathcal{P}_2 = \langle Q_2, q_{0,2}, \Delta_2 \rangle$ and $\mathcal{P}_3 = \langle Q_3, q_{0,3}, \Delta_3 \rangle$ be defined as we just informally described, with a slight complication about the initial state of $\mathcal{P}_2$ (this is motivated by technical reasons that will become clear soon).

$$
\begin{aligned}
Q_2 &= \{q_{0,2}, q_{1,2}\} \cup \{q_{a,1}, q_{a,2} \mid a \in \Sigma\} \qquad Q_3 = \{q_{0,3}\} \cup \{q_{a,1}, q_{a,2}, q_{a,3} \mid a \in \Sigma\} \\
\Delta_2 &= \{(q_{0,2}, ?a^{3\to2}, q_{a,1}), (q_{1,2}, ?a^{3\to2}, q_{a,1}), (q_{a,1}, ?a^{1\to2}, q_{a,2}), (q_{a,2}, !a^{2\to3}, q_{1,2}) \mid a \in \Sigma\} \\
\Delta_3 &= \{(q_{0,3}, ?a^{1\to3}, q_{a,1}), (q_{a,1}, !a^{3\to2}, q_{a,2}), (q_{a,2}, ?a^{2\to3}, q_{a,3}), (q_{a,3}, !a^{3\to1}, q_{0,3}) \mid a \in \Sigma\}
\end{aligned}
$$

▶ **Example 5.** Consider $\Sigma = \{a, \underline{m}\}$ and the FIFO automaton $\mathcal{A} = \langle \{q_0, q_1\}, \mathsf{Act}_\Sigma, \Delta, q_0 \rangle$ with transition relation $\Delta_\mathcal{A} = \{(q_0, !a, q_0), (q_0, !\underline{m}, q_1), (q_1, ?a, q_0), (q_1, ?\underline{m}, q_0)\}$. Then $\mathcal{A}$ and the peers $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ are depicted in Fig. 2.

Let $\mathcal{S}_\mathcal{A} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$. There is a tight correspondence between the k-bounded traces of $\mathcal{A}$, for $k \geq 1$, and the k-bounded traces of $\mathcal{S}_\mathcal{A}$: every trace $\tau \in \mathsf{Traces}_k(\mathcal{A})$ induces the trace $h(\tau) \in \mathsf{Traces}_k(\mathcal{S}_\mathcal{A})$ where $h : \mathsf{Act}_\Sigma^* \to \mathsf{Act}_M$ is the homomorphism from the traces of $\mathcal{A}$ to the traces of $\mathcal{S}_\mathcal{A}$ defined by $h(!a) = !a^{1\to2}$ and $h(?a) = !?a^{1\to3} \cdot !?a^{3\to2} \cdot ?a^{1\to2} \cdot !?a^{2\to3} \cdot !?a^{3\to1}$. The converse is not true: there are traces of $\mathcal{S}_\mathcal{A}$ that are not prefixes of a trace $h(\tau)$ for some $\tau \in \mathsf{Traces}_k(\mathcal{A})$. This happens when $\mathcal{P}_1$ sends an order to dequeue $a^{1\to3}$ that correspond to a transition $?a$ that $\mathcal{A}$ cannot execute. In that case, the system blocks when $\mathcal{P}_2$ has to execute the order.

▶ **Lemma 6.** *For all $k \geq 0$,* $\mathsf{Traces}_k(\mathcal{S}_\mathcal{A}) = \downarrow \{h(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{A})\} \cup \downarrow \{h(\tau) \cdot !?a^{1\to3} \cdot !?a^{3\to2} \mid \tau \in \mathsf{Traces}_k(\mathcal{A}), (q_0, \epsilon) \xrightarrow{\tau} (q, w), (q, ?a, q') \in \Delta\}$.

Since $\mathcal{A}$ satisfies $(R1)$, all stable configurations that are reachable in $\mathcal{S}_\mathcal{A}$ are reachable by a synchronous trace, and since it satisfies $(R2)$, the only reachable stable configuration is the initial configuration. Moreover, $\mathcal{J}_0(\mathcal{S}_A) = \{\epsilon\}$ and $\mathcal{J}_k(\mathcal{S}_\mathcal{A}) \neq \{\epsilon\}$ for $k \geq 1$ (provided $\mathcal{A}$ sends at least one message). As a consequence, $\mathcal{S}_\mathcal{A}$ is not synchronizable.

Let us fix now a special message $\underline{m} \in \Sigma$. We would like to turn $\mathcal{S}_\mathcal{A}$ into a system that is synchronizable, except for the send traces that contain $\underline{m}^{2\to3}$. Note that, by Lemma 6, $\mathcal{S}_\mathcal{A}$

has a send trace that contains $\underline{m}^{2\to3}$ if and only if there are traces of $\mathcal{A}$ that contain $?\underline{m}$. Roughly, we need to introduce new behaviours for the peer 2 that will "flood" the system with many synchronous traces. Let $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3\rangle$ be the system $\mathcal{S}_\mathcal{A}$ in which the peer $\mathcal{P}_2$ is replaced with the peer $\mathcal{P}'_2 = \langle Q'_2, q_{0,2}, \Delta'_2\rangle$ defined as follows.

$$\begin{array}{rcl} Q'_2 & = & \{q_{0,2}, q'_{0,2}\} \cup \{q'_{a,1} \mid a \in \Sigma, a \neq \underline{m},\} \cup \{q_\perp\} \\ \Delta'_2 & = & \{(q_{0,2}, ?a^{1\to2}, q'_{0,2}), (q, ?a^{1\to2}, q) \mid a \in \Sigma, q \neq q_{0,2}\} \\ & \cup & \{(q_{0,2}, ?a^{3\to2}, q'_{a,1}), (q'_{0,2}, ?a^{3\to2}, q'_{a,1}), (q'_{a,1}, !a^{2\to3}, q'_{0,2}), \mid a \in \Sigma, a \neq \underline{m}\} \\ & \cup & \{(q, ?\underline{m}^{3\to2}, q_\perp) \mid q \in Q'_2\} \end{array}$$

▶ **Example 7.** For $\Sigma = \{a, \underline{m}\}$, and $\mathcal{A}$ as in Example 5, $\mathcal{P}'_2$ is depicted in Fig. 2 (omitting the transitions to the sink state $q_\perp$).

Intuitively, $\mathcal{P}'_2$ can always receive any message from peer $\mathcal{P}_1$. Like $\mathcal{P}_2$, it can also receive orders to dequeue from peer $\mathcal{P}_3$, but instead of executing the order before sending an acknowledgement, it ignores the order as follows. If $\mathcal{P}'_2$ receives the order to dequeue a message $a^{1\to2} \neq \underline{m}^{1\to2}$, $\mathcal{P}'_2$ acknowledges $\mathcal{P}_3$ but does not dequeue in the $1 \to 2$ queue. If the order was to dequeue $\underline{m}$, $\mathcal{P}'_2$ blocks in the sink state $q_\perp$. The system $\mathcal{S}'_\mathcal{A} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3\rangle$ contains many synchronous traces: any M-trace $\tau \in L(\mathcal{P}_1)$ labeling a path in automaton $\mathcal{P}_1$ can be lifted to a synchronous trace $\tau' \in \mathsf{Traces}_0(\mathcal{S}_{\mathcal{A},\underline{m}})$ provided $!\underline{m}^{1\to3}$ does not occur in τ. However, if $\mathcal{P}_1$ takes a $!\underline{m}^{1\to3}$ transition, it gets blocked for ever waiting for $\underline{m}^{3\to1}$. Therefore, if $!a^{1\to3}$ occurs in a synchronous trace τ of $\mathcal{S}'_{\mathcal{A},\underline{m}}$, it must be in the last four actions, and this trace leads to a deadlock configuration in which both 1 and 3 wait for an acknowledgement and 2 is in the sink state.

Let $L^{\underline{m}}(\mathcal{A})$ be the set of traces τ recognized by $\mathcal{A}$ as a finite state automaton (over the alphabet Act_Σ) such that either $?\underline{m}$ does not occur in τ, or it occurs only once and it is the last action of τ. For instance, with $\mathcal{A}$ as in Example 5, $L^{\underline{m}}(\mathcal{A}) =\downarrow \left(!a^* \cdot !\underline{m} \cdot ?a\right)^* \cdot !a^* \cdot !\underline{m} \cdot ?\underline{m}$. Let $h' : \mathsf{Act}^*_\Sigma \to \mathsf{Act}^*_M$ be the morphism defined by $h'(!a) =!?a^{1\to2}$ for all $a \in \Sigma$, $h'(?a) = !?a^{1\to3} \cdot !?a^{3\to2} \cdot !?a^{2\to3} \cdot !?a^{3\to1}$ for all $a \neq \underline{m}$, and $h'(?\underline{m}) =!?\underline{m}^{1\to3} \cdot !?\underline{m}^{3\to2}$.

▶ **Lemma 8.** $\mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}}) =\downarrow \{h'(\tau) \mid \tau \in L^{\underline{m}}(\mathcal{A})\}$.

Let us now consider an arbitrary trace $\tau \in \mathsf{Traces}_\omega(\mathcal{S}'_{\mathcal{A},\underline{m}})$. Let $h'' : \mathsf{Act}^*_M \to \mathsf{Act}^*_M$ be such that $h''(!a^{1\to2}) =!?a^{1\to2}$, $h''(?a^{1\to2}) = \epsilon$, and $h''(\lambda) = \lambda$ otherwise. Then $h''(\tau) \in \mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$ and $\tau \overset{\mathcal{S}}{\sim} h''(\tau)$ for $\mathcal{S} = \mathcal{S}'_{\mathcal{A},\underline{m}}$. Indeed, τ and $h''(\tau)$ are the same up to insertions and deletions of receive actions $?a^{1\to2}$, and every state of $\mathcal{P}'_2$ (except the initial one) has a self loop $?a^{1\to2}$. Therefore,

▶ **Lemma 9.** *$\mathcal{S}'_{\mathcal{A},\underline{m}}$ is synchronizable.*

Let us now consider the system $\mathcal{S}''_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}_2 \cup \mathcal{P}'_2, \mathcal{P}_3\rangle$, where $\mathcal{P}_2 \cup \mathcal{P}'_2 = \langle Q_2 \cup Q'_2, q_{02}, \Delta_2 \cup \Delta''_2\rangle$ is obtained by merging the initial state $q_{0,2}$ of $\mathcal{P}_2$ and $\mathcal{P}'_2$. Note that $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_k(\mathcal{S}_\mathcal{A}) \cup \mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}})$, because $q_{0,2}$ has no incoming edge in $\mathcal{P}_2 \cup \mathcal{P}'_2$.

▶ **Lemma 10.** *Let $k \geq 1$. The following two are equivalent:*
1. *there is τ such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}_k(\mathcal{A})$;*
2. *$\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) \neq \mathcal{I}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$.*

Proof. Let $k \geq 1$ be fixed.

(1) $\Longrightarrow$ (2) Let τ be such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}_k(\mathcal{A})$. By Lemma 6, there is $\upsilon \in \mathcal{I}_k(\mathcal{S}_\mathcal{A})$ such that $\underline{m}^{2\to3}$ occurs in υ (take $\upsilon = \mathsf{send}(h(\tau \cdot ?\underline{m}))$). By Lemma 6, $\upsilon \notin \mathcal{I}_0(\mathcal{S}_\mathcal{A}) = \emptyset$, and by Lemma 8, $\upsilon \notin \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. Therefore $\upsilon \in \mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) \setminus \mathcal{I}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$.

(2) $\Longrightarrow$ (1) By contraposite. Let $\mathsf{Traces}_k(\mathcal{A}\backslash ?\underline{m}) = \{\tau \in \mathsf{Traces}_k(\mathcal{A}) \mid ?\underline{m} \text{ does not occur in } \tau\}$, and let us assume $\neg(1)$, *i.e.* $\mathsf{Traces}_k(\mathcal{A}\backslash ?\underline{m}) = \mathsf{Traces}_k(\mathcal{A})$. Let us show that $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$. From the assumption $\neg(1)$ and Lemma 6, it holds that $\mathsf{Traces}_k(\mathcal{S}_{\mathcal{A}}) =$

$$\begin{array}{rl} & \downarrow\{h(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{A}\backslash ?\underline{m})\} \\ \cup & \downarrow\{h(\tau)\cdot !?a^{1\to 3}\cdot !?a^{3\to 2} \mid \tau \in \mathsf{Traces}_k(\mathcal{A}\backslash ?\underline{m}), (q_0,\epsilon) \xrightarrow{\tau} (q,w), (q,?a,q') \in \Delta\}. \end{array}$$

By $\mathsf{send}(h(\tau)) = \mathsf{send}(h'(\tau))$ and $\mathsf{Traces}_k(\mathcal{A}\backslash ?\underline{m}) \subseteq L^{\underline{m}}(\mathcal{A})$, we get that

$$\mathcal{I}_k(\mathcal{S}_{\mathcal{A}}) \quad \subseteq \quad \downarrow\{\mathsf{send}(h'(\tau)) \mid \tau \in L^{\underline{m}}(\mathcal{A})\}$$

and therefore, by Lemma 8, $\mathcal{I}_k(\mathcal{S}_{\mathcal{A}}) \subseteq \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. Since $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_k(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}})$ and since by Lemma 9 $\mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}}) = \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$, we get that $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) \subseteq \mathcal{I}_0(\mathcal{S}''_{\mathcal{A}\underline{m}})$, and thus $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_0(\mathcal{S}''_{\mathcal{A}\underline{m}})$. ◀

▶ **Theorem 11.** *Synchronizability (resp. language synchronizability) is undecidable.*

Proof. Let a FIFO automaton $\mathcal{A}$ satisfying $(R1)$ and $(R2)$ and a message $\underline{m}$ be fixed. By Lemma 10, $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is non synchronizable iff there is a trace τ such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}_\omega(\mathcal{A})$. By Lemma 4, this is an undecidable problem. ◀

4 The case of oriented rings

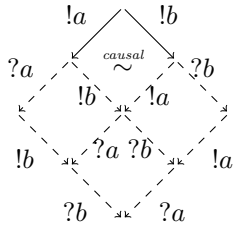
In the previous section we established the undecidability of synchronizability for systems with (at least) three peers. In this section, we show that this result is tight, in the sense that synchronizability is decidable if G_M is an oriented ring, in particular if the system involves two peers only. This relies on the fact that 1-synchronizability implies synchronizability for such systems. This property is highly non-trivial, and below we only sketch the main steps of the proof, identifying when the hypothesis on the ring topology becomes necessary.

The starting point is a confluence property on arbitrary 1-synchronizable systems.

▶ **Lemma 12.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a, b \in M$ be such that*

1. $\tau \cdot !a \in \mathsf{Traces}_1(\mathcal{S})$,
2. $\tau \cdot !b \in \mathsf{Traces}_1(\mathcal{S})$, *and*
3. $\mathsf{src}(a) \neq \mathsf{src}(b)$.

If υ_1, υ_2 are any two of the six different shuffles of $!a \cdot ?a$ with $!b \cdot ?b$, then $\tau \cdot \upsilon_1 \in \mathsf{Traces}_\omega(\mathcal{S})$, $\tau \cdot \upsilon_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ and $\tau \cdot \upsilon_1 \overset{\mathcal{S}}{\sim} \tau \cdot \upsilon_2$.



▶ Remark. This lemma should not be misunderstood as a consequence of causal equivalence. Observe indeed that the square on top of the diagram is the only square that commutes for causal equivalence. The three other squares only commute with respect to $\overset{\mathcal{S}}{\sim}$, and they commute for $\overset{\text{causal}}{\sim}$ only if some extra assumptions on a and b are made. For instance, the left square does commute for $\overset{\text{causal}}{\sim}$ if and only if $\mathsf{dst}(a) \neq \mathsf{src}(b)$.

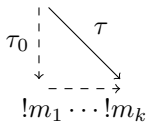
Lemma 12 then generalizes to arbitrary sequences of send actions with rather technical arguments.

▶ **Lemma 13.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $a_1, \ldots, a_n, b_1, \ldots b_m \in M$ and $\tau \in \mathsf{Traces}_0(\mathcal{S})$ be such that*
1. $\tau \cdot !a_1 \cdots !a_n \in \mathsf{Traces}_n(\mathcal{S})$,
2. $\tau \cdot !b_1 \cdots !b_m \in \mathsf{Traces}_m(\mathcal{S})$, *and*
3. $\mathsf{src}(a_i) \neq \mathsf{src}(b_j)$ *for all* $i \in \{1, \ldots, n\}$ *and* $j \in \{1, \ldots, m\}$.
Then for any two different shuffles υ_1, υ_2 of $!?a_1 \cdot !?a_2 \cdots !?a_n$ with $!?b_1 \cdot !?b_2 \cdots !?b_m$, it holds that $\tau \cdot \upsilon_1 \in \mathsf{Traces}_\omega(\mathcal{S})$, $\tau \cdot \upsilon_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ and $\tau \cdot \upsilon_1 \overset{\mathcal{S}}{\sim} \tau \cdot \upsilon_2$.

For the rest of the proof, the hypothesis on the communication topology being an oriented ring becomes necessary. We follow the rough idea in [4], also used for half-duplex systems [7], and show a trace normalization property.

▶ **Definition 14** (Normalized trace)**.** A M-trace τ is *normalized* if there is a synchronous M-trace τ_0, $n \geq 0$, and messages $a_1, \ldots, a_n$ such that $\tau = \tau_0 \cdot !a_1 \cdots !a_n$.

▶ **Lemma 15** (Trace Normalization)**.** *Assume M is such that the communication topology G_M is an oriented ring. Let $\mathcal{S} = \langle \mathcal{P}_1, \ldots, \mathcal{P}_p \rangle$ be a 1-synchronizable M-system. For all $\tau \in \mathsf{Traces}_\omega(\mathcal{S})$, there is a normalized trace $\mathsf{norm}(\tau) \in \mathsf{Traces}_\omega(\mathcal{S})$ such that $\tau \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau)$.*



Proof. By induction on τ. Let $\tau = \tau' \cdot \lambda$, be fixed. Let us assume by induction hypothesis that there is a normalized trace $\mathsf{norm}(\tau') \in \mathsf{Traces}_\omega(\mathcal{S})$ such that $\tau' \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau')$. Let us reason by case analysis on the last action λ of τ. The easy case is when λ is a send action: then, $\mathsf{norm}(\tau') \cdot \lambda$ is a normalized trace, and $\mathsf{norm}(\tau') \cdot \lambda \overset{\mathcal{S}}{\sim} \tau' \cdot \lambda$ by right congruence of $\overset{\mathcal{S}}{\sim}$. The difficult case is when λ is $?a$ for some $a \in M$. Let $i = \mathsf{src}(a)$, $j = \mathsf{dst}(a)$, *i.e.* $i + 1 = j \bmod p$. By the definitions of a normal trace and $\overset{\text{causal}}{\sim}$, there are $\tau'_0 \in \mathsf{Traces}_0(\mathcal{S})$, $a_1, \ldots, a_n, b_1, \ldots, b_m \in M$ such that

$$\mathsf{norm}(\tau') \overset{\text{causal}}{\sim} \tau'_0 \cdot !a_1 \cdots !a_n \cdot !b_1 \cdots !b_m$$

with $\mathsf{src}(a_k) = i$ for all $k \in \{1, \ldots, n\}$ and $\mathsf{src}(b_k) \neq i$ for all $k \in \{1, \ldots, m\}$. Since G_M is an oriented ring, $\mathsf{dst}(a_1) = j$, therefore $a_1 = a$ (because by hypothesis j may receive a in the configuration that $\mathsf{norm}(\tau')$ leads to). Let $\mathsf{norm}(\tau) = \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_m \cdot !a_2 \cdots !a_n$ and let us show that $\mathsf{norm}(\tau) \in \mathsf{Traces}_\omega(\mathcal{S})$ and $\tau \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau)$.

Since $\mathsf{norm}(\tau') \in \mathsf{Traces}_\omega(\mathcal{S})$, we have in particular that $\tau'_0 \cdot !a \in \mathsf{Traces}_1(\mathcal{S})$ and $\tau'_0 \cdot !b_1 \cdots !b_n \in \mathsf{Traces}_\omega(\mathcal{S})$. Consider the two traces

$$\begin{aligned} \upsilon_1 = &\quad \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \cdot ?b_1 \cdots ?b_n \\ \upsilon_2 = &\quad \tau'_0 \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \cdot ?b_1 \cdots ?b_n. \end{aligned}$$

By Lemma 13, $\upsilon_1, \upsilon_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ and both lead to the same configuration, and in particular to the same control state q for peer j. The actions $?b_1$, $?b_2, \ldots ?b_n$ are not executed by peer j (because $\mathsf{src}(m) \neq i$ implies $\mathsf{dst}(m) \neq j$ on an oriented ring), so the two traces

$$\begin{aligned} \upsilon'_1 = &\quad \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \\ \upsilon'_2 = &\quad \tau'_0 \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \end{aligned}$$

lead to two configurations γ'_1, γ'_2 with the same control state q for peer j as in the configuration reached after υ_1 or υ_2. On the other hand, for all $k \neq j$, $\mathsf{onPeer}_k(\upsilon'_1) = \mathsf{onPeer}_k(\upsilon'_2)$, therefore $\upsilon'_1 \overset{\mathcal{S}}{\sim} \upsilon'_2$. Since $\tau'_0 \cdot !a \cdot !a_2 \cdots !a_n \in \mathsf{Traces}_n(\mathcal{S})$, and $\mathsf{onPeer}_i(\tau'_0 \cdot !a) = \mathsf{onPeer}_i(\upsilon'_1) = \mathsf{onPeer}_i(\upsilon'_2)$, the two traces

$$\begin{aligned} \upsilon''_1 = &\quad \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \cdot !a_2 \cdots !a_n \\ \upsilon''_2 = &\quad \tau'_0 \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \cdot !a_2 \cdots !a_n \end{aligned}$$

belong to $\mathsf{Traces}_\omega(\mathcal{S})$ and $\upsilon''_1 \overset{\mathcal{S}}{\sim} \upsilon''_2$. Consider first υ''_1: this is $\mathsf{norm}(\tau)$ as defined above, therefore $\mathsf{norm}(\tau) \in \mathsf{Traces}_\omega(\mathcal{S})$, and $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \upsilon''_2$. Consider now υ''_2. By definition, $\upsilon''_2 \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot ?a$. By hypothesis, $\mathsf{norm}(\tau') \overset{\mathcal{S}}{\sim} \tau'$, therefore $\mathsf{norm}(\tau') \cdot ?a \overset{\text{causal}}{\sim} \tau$. To sum up, $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \upsilon''_2 \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot ?a \overset{\mathcal{S}}{\sim} \tau$, therefore $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \tau$. ◀

As a consequence of Lemma 15, 1-synchronizability implies several interesting properties on the reachability set.

▶ **Definition 16** (Channel-recognizable reachability set [17, 7])**.** Let $\mathcal{S} = \langle \mathcal{P}_1, \ldots, \mathcal{P}_p \rangle$ with $\mathcal{P}_i = \langle Q_i, \Delta_i, q_{0,i} \rangle$. The (coding of the) *reachability set* of $\mathcal{S}$ is the language $\mathsf{Reach}(\mathcal{S})$ over the alphabet $(M \cup \bigcup_{i=1}^{p} Q_i)^*$ defined as $\{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma_0 \xrightarrow{\tau} (q_1, \ldots, q_p, w_1, \ldots, w_p), \tau \in \mathsf{Traces}_\omega(\mathcal{S})\}$. $\mathsf{Reach}(\mathcal{S})$ is *channel-recognizable* (or QDD representable [5]) if it is a recognizable (and rational) language.

▶ **Theorem 17.** *Let M be a message set such that G_M is an oriented ring, and let $\mathcal{S}$ be a M-system that is 1-synchronizable. Then*

1. *the reachability set of $\mathcal{S}$ is channel recognizable,*
2. *for all $\tau \in \mathsf{Traces}_\omega(\mathcal{S})$, for all $\gamma_0 \xrightarrow{\tau} \gamma$, there is a stable configuration γ', $n \geq 0$ and $m_1, \ldots m_n \in M$ such that $\gamma \xrightarrow{?m_1 \cdots ?m_n} \gamma'$.*

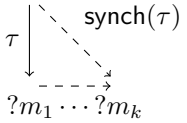
In particular, $\mathcal{S}$ neither has orphan messages nor unspecified receptions [7].

Proof. 1. Let S be the set of stable configurations γ such that $\gamma_0 \xrightarrow{\tau} \gamma$ for some $\tau \in \mathsf{Traces}_0(\mathcal{S})$; S is finite and effective. By Lemma 15, $\mathsf{Reach}(\mathcal{S}) = \bigcup\{\mathsf{Reach}^!(\gamma) \mid \gamma \in S\}$, where $\mathsf{Reach}^!(\gamma) = \{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma \xrightarrow{!a_1 \cdots !a_n} (q_1, \ldots, q_p, w_1, \ldots, w_p), n \geq 0, a_1, \ldots a_n \in M\}$ is an effective rational language.

2. Assume $\gamma_0 \xrightarrow{\tau} \gamma$. By Lemma 15, $\gamma_0 \xrightarrow{\tau_0 \cdot !m_1 \cdots !m_r} \gamma$ for some $\tau_0 \in \mathsf{Traces}_0(\mathcal{S})$. Then $\tau_0 \cdot !m_1 \cdots !m_r \overset{\text{causal}}{\sim} \tau_0 \cdot \tau_1$ where $\tau_1 := !a_1 \cdots !a_n \cdot b_1 \cdots b_m$ for some $a_1, \ldots, a_n, b_1, b_m$ such that $\mathsf{src}(a_i) \neq \mathsf{src}(b_j)$ for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. By Lemma 13, $\tau_0 \cdot \tau_1 \cdot \overline{\tau_1} \in \mathsf{Traces}_\omega(\mathcal{S})$ (where $\overline{\tau_1} = ?a_1 \cdots ?a_n \cdot ?b_1 \cdots ?b_m$), and therefore $\gamma_0 \xrightarrow{\tau_0 \cdot \tau_1} \gamma \xrightarrow{\overline{\tau_1}} \gamma'$ for some stable configuration γ'.

◀

▶ **Theorem 18.** *Let M be a message set such that G_M is an oriented ring. For any M-system $\mathcal{S}$, $\mathcal{S}$ is 1- synchronizable if and only if it is synchronizable.*



In order to prove Theorem 18, we prove by induction on the length of τ that $\tau \cdot ?m_1 \cdots ?m_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau)$ for some messages $m_1, \ldots, m_k$, where $\mathsf{synch}(\tau)$ denotes the unique synchronous M-trace such that $\mathsf{send}(\mathsf{synch}(\tau)) = \mathsf{send}(\tau)$.

▶ **Theorem 19.** *Let M be a message set such that G_M is an oriented ring. The problem of deciding whether a given M-system is synchronizable is decidable.*

5 Extensions

We considered the notions of synchronizability and language synchronizability introduced by Basu and Bultan [2] and we showed that both are not decidable for systems with peer-to-peer FIFO communications, called (1-1) type systems in [2]. In the same work, Basu and Bultan considered the question of the decidability of language synchronizability for other communication models. All the results we presented so far do not have any immediate consequences on their claims for these communication models. Therefore, we briefly discuss now what we can say about the decidability of language synchronizability for the other communication models that have been considered.

Non FIFO communications. In [2], language synchronizability is studied for systems where peers communicate through bags instead of queues, thus allowing to reorder messages. Language synchronizability is decidable for bag communications: $\mathsf{Traces}_\omega^{bag}(\mathcal{S})$ is the trace language of a Petri net, $T_0(\mathcal{S}) = \{\tau \in \mathsf{Act}_M^* \mid \mathsf{send}(\tau) \in \mathcal{J}_0^{bag}(\mathcal{S})\}$ is an effective regular language, $\mathcal{S}$ is language synchronizable iff $\mathsf{Traces}_\omega^{bag}(\mathcal{S}) \subseteq T_0(\mathcal{S})$, and whether the trace language of a Petri is included in a given regular language reduces to the coverability problem. Lossy communications where not considered in [2], but the same kind of argument would also hold for lossy communications. However, our Example 1 is a counter-example for Lemma 3 in [2], *i.e.* the notion of language 1-synchronizability for bag communications defined in [2] does not imply language synchronizability. The question whether (language) synchronizability can be decided more efficiently than by reduction to the coverability problem for Petri nets is open.

Non peer-to-peer communications. The other communication models considered in [2] keep the FIFO queue model, but differ in the way queues are distributed among peers. The $*$-1 (mailbox) model assumes a queue per receiver. This model is the first model that was considered for (language) synchronizability [1, 4]. Our Example 1 is not easy to adapt for this communication model. We therefore design a completely different counter-example.

▶ **Example 20.** Consider the system of communicating machines depicted in Fig. 3. Assume that the machines communicate via mailboxes, like in [1, 4], i.e. all messages that are sent to peer i wait in a same FIFO queue Q_i, and let $\mathcal{J}_k^{*-1}(\mathcal{S})$ denote the k-bounded send traces of $\mathcal{S}$ within this model of communications. Then $\mathcal{J}_0^{*-1}(\mathcal{S}) = \mathcal{J}_1^{*-1}(\mathcal{S}) \neq \mathcal{J}_2^{*-1}(\mathcal{S})$, as depicted in Figure 3. Therefore $\mathcal{S}$ is language 1-synchronizable but not language synchronizable, which contradicts Theorem 1 in [1], Theorem 2 in [3], and Theorem 2 in [2]. It can be noticed that it does not contradict Theorem 1 in [4], but it contradicts the Lemma 1 of the same paper, which is used to prove Theorem 1.

Many problems are therefore left for future work: the (un)decidability of synchronizability for the mailbox semantics, the largest class of communication topologies for which 1-synchronizability implies synchronizability (either for the peer-to-peer semantics or for the mailbox one), the study of language synchronizability, etc. Our intention in this work was limited to the identification of some of these problems, and maybe to explain why the errors in [1, 4, 2] were missed by so many reviewers.

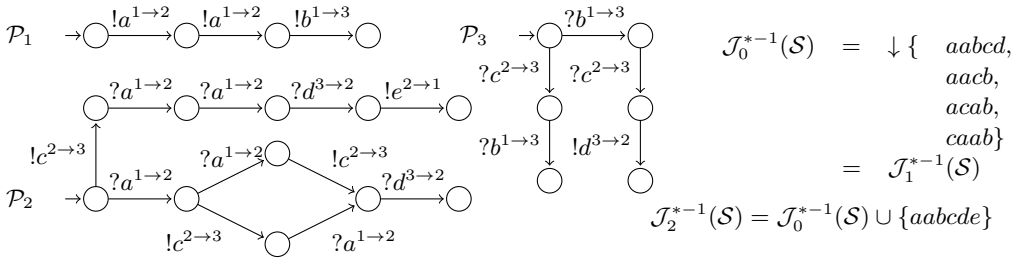


Figure 3 Language 1-synchronizability does not imply language synchronizability for 1-$*$ (mailbox) communications *à la* [1, 4].

Acknowledgement. We would like to thank the anonymous reviewers of our paper for relevant suggestions of improvements, and for an accurate reading of our proofs.

References

1 Samik Basu and Tevfik Bultan. Choreography conformance via synchronizability. In *Procs. of WWW 2011*, pages 795–804, 2011. doi:10.1145/1963405.1963516.

2 Samik Basu and Tevfik Bultan. On deciding synchronizability for asynchronously communicating systems. *Theor. Comput. Sci.*, 656:60–75, 2016. doi:10.1016/j.tcs.2016.09.023.

3 Samik Basu, Tevfik Bultan, and Meriem Ouederni. Deciding choreography realizability. In *Procs. of POPL'12*, pages 191–202, 2012. doi:10.1145/2103656.2103680.

4 Samik Basu, Tevfik Bultan, and Meriem Ouederni. Synchronizability for verification of asynchronously communicating systems. In *Procs. of VMCAI 2012*, 2012. doi:10.1007/978-3-642-27940-9_5.

5 Bernard Boigelot and Patrice Godefroid. Symbolic verification of communication protocols with infinite state spaces using qdds. *Formal Methods in System Design*, 14(3):237–255, 1999. doi:10.1023/A:1008719024240.

6 Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, April 1983. doi:10.1145/322374.322380.

7 Gerald Cécé and Alain Finkel. Verification of programs with half-duplex communication. *Inf. Comput.*, 202(2):166–190, 2005. doi:10.1016/j.ic.2005.05.006.

8 Pierre Chambart and Philippe Schnoebelen. Mixing lossy and perfect fifo channels. In *Procs. of CONCUR 2008*, pages 340–355, 2008. doi:10.1007/978-3-540-85361-9_28.

9 Lorenzo Clemente, Frédéric Herbreteau, and Grégoire Sutre. Decidable topologies for communicating automata with FIFO and bag channels. In *Procs. of CONCUR 2014*, pages 281–296, 2014. doi:10.1007/978-3-662-44584-6_20.

10 Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata. In *Procs. of ESOP 2012*, pages 194–213, 2012. doi:10.1007/978-3-642-28869-2_10.

11 Alain Finkel and Etienne Lozes. Synchronizability of communicating finite state machines is not decidable. Technical report, arXiv, 2017. URL: https://arxiv.org/abs/1702.07213.

12 Blaise Genest, Dietrich Kuske, and Anca Muscholl. A kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006. doi:10.1016/j.ic.2006.01.005.

13 Alexander Heußner, Tristan Le Gall, and Grégoire Sutre. Mcscm: A general framework for the verification of communicating machines. In *Procs. of TACAS 2012*, pages 478–484, 2012. doi:10.1007/978-3-642-28756-5_34.

14 Alexander Heußner, Jérôme Leroux, Anca Muscholl, and Grégoire Sutre. Reachability analysis of communicating pushdown systems. *Logical Methods in Computer Science*, 8(3), 2012. doi:10.2168/LMCS-8(3:23)2012.

15 Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. Context-bounded analysis of concurrent queue systems. In *Procs. of TACAS 2008*, pages 299–314, 2008. doi:10.1007/978-3-540-78800-3_21.

16 Rajit Manohar and Alain J. Martin. Slack elasticity in concurrent computing. In *Procs. of Math. of Prog. Construction (MPC'98)*, pages 272–285, 1998. doi:10.1007/BFb0054295.

17 Jan Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. of Protocol Specification, Testing, and Verification, VII*, 1987.

18 Stephen F. Siegel. Efficient verification of halting properties for MPI programs with wildcard receives. In *Procs. of VMCAI 2005*, pages 413–429, 2005. doi:10.1007/978-3-540-30579-8_27.

19 Sarvani Vakkalanka, Anh Vo, Ganesh Gopalakrishnan, and Robert M. Kirby. Precise dynamic analysis for slack elasticity: Adding buffering without adding bugs. In *Procs. of EuroMPI 2010*, pages 152–159, 2010. doi:10.1007/978-3-642-15646-5_16.

Admissibility in Concurrent Games*†

Nicolas Basset[1], Gilles Geeraerts[2], Jean-François Raskin[3], and Ocan Sankur[4]

1 **Université libre de Bruxelles, Brussels, Belgium**
nicolas.basset@ulb.ac.be
2 **Université libre de Bruxelles, Brussels, Belgium**
gilles.geeraerts@ulb.ac.be
3 **Université libre de Bruxelles, Brussels, Belgium**
jraskin@ulb.ac.be
4 **CNRS, IRISA, Rennes, France**
ocan.sankur@irisa.fr

Abstract

In this paper, we study the notion of *admissibility* for randomised strategies in *concurrent games*. Intuitively, an admissible strategy is one where the player plays 'as well as possible', because there is no other strategy that *dominates* it, i.e., that wins (almost surely) against a superset of adversarial strategies. We prove that admissible strategies always exist in concurrent games, and we characterise them precisely. Then, when the objectives of the players are ω-regular, we show how to perform *assume-admissible synthesis*, i.e., how to compute admissible strategies that win (almost surely) under the hypothesis that the other players play admissible strategies only.

1998 ACM Subject Classification D.2.4 Software/Program Verification, F.3.1 Specifying and Verifying and Reasoning about Programs, I.2.2 Program Synthesis

Keywords and phrases Multi-player games, admissibility, concurrent games, randomized strategies

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.123

1 Introduction

In a concurrent n-player game played on a graph, all n players *independently* and *simultaneously* choose moves at each round of the game, and those n choices determine the next state of the game [14]. Concurrent games generalise turn-based games and it is well-known that, while deterministic strategies are sufficient in the turn-based case, randomised strategies are necessary for winning with probability one even for reachability objectives. Intuitively, randomisation is necessary because, in concurrent games, in each round, players choose their moves simultaneously. Randomisation makes it possible to choose a good move with some probability without the knowledge of the moves that the other players are simultaneously choosing. As a consequence, there are two classical semantics that are considered to analyse these games *qualitatively*: winning with certainty (sure semantics in the terminology of [14]), and winning with probability one (almost sure semantics in the terminology of [14]). We consider both semantics here.

* An extended version of this article is available in [4], http://arxiv.org/abs/1702.06439.

† This work was partially supported by the ERC Starting grant 279499 (inVEST), the ARC project « Non-Zero Sum Game Graphs: Applications to Reactive Synthesis and Beyond » (Fédération Wallonie-Bruxelles), J.-F. Raskin is Professeur Francqui de Recherche.

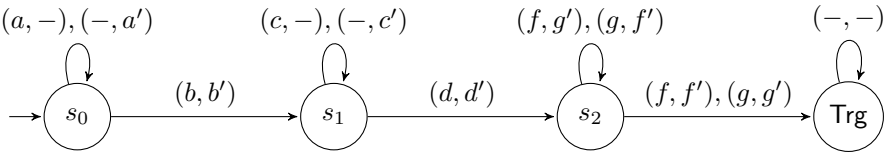
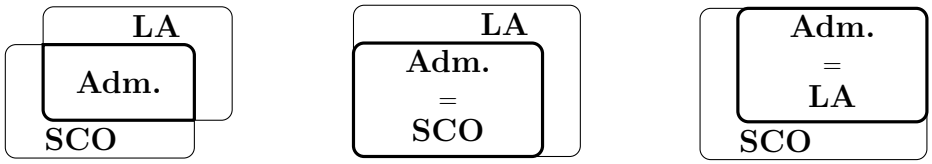


Figure 1 A concurrent game where Player 1 and 2 want to reach Trg and s_2 respectively.



(a) Concurrent games. **(b)** Turn-based games. **(c)** Safety games.

Figure 2 The relationships between the classes of Admissible, LA, and SCO strategies for three families of games. All the inclusions are strict.

Previous papers on concurrent games are mostly concerned with two-player zero-sum games, i.e. two players that have fully antagonistic objectives. In this paper, we consider the *more general setting* of n-player non zero-sum concurrent games in which each player has its own objective. The notion of winning strategy is not sufficient to study non zero-sum games and other solution concepts have been proposed. One such concept is the notion of *admissible strategy* [1].

For a player with objective Φ, a strategy σ is said to be *dominated* by a strategy σ' if σ' does as well as σ with respect to Φ against all the strategies of the other players and strictly better for some of them. A strategy σ is *admissible* for a player if it is not dominated by any other of his strategies. Clearly, playing a strategy which is not admissible is sub-optimal and a rational player should only play admissible strategies. While recent works have studied the notion of admissibility for n-player non zero-sum game graphs [5, 15, 10, 8, 7], they are all concerned with the special case of turn-based games and this work is the first to consider the more general concurrent games.

Throughout the paper, we consider the running example in Figure 1. This is a concurrent game played by two players. Player 1's objective is to reach Trg, while Player 2 wants to reach s_2. Edges are labelled by pairs of moves of both players which activate that transition (where $-$ means 'any move'). It is easy to see that no player can enforce its objective with or without randomisation, so, there is no winning strategy in this game for either player. This is because moving from s_0 to s_1 and from s_1 to s_2 requires the cooperation of both players. Moreover, the transitions from s_2 behave as in the classical 'matching pennies' game: player 1 must chose between f and g; player 2 between f' and g'; and the target is reached only when the choices 'match'. So, randomisation is needed to make sure Trg is reached with probability one, from s_2. In the paper, we will describe the dominated and admissible strategies of this game.

Technical contributions. First, we study the notion of admissible strategies for both the *sure* and *almost sure* semantics of concurrent games. We show in Theorem 8 that in both semantics admissible strategies always exist. The situation is thus similar to the turn-based case [5, 10]. Nevertheless, the techniques used in this simpler case do not generalise easily to the concurrent case and we need substantially more involved technical tools here. To obtain

our universal existence result, we introduce two weaker solution concepts: *locally admissible moves* and *strongly cooperative optimal* strategies. While cooperative optimal strategies were already introduced in [7] and shown equivalent to admissible strategies in the turn based setting, they are strictly weaker than admissible strategies in the concurrent setting (both for the sure and the almost sure semantics), and they need to be combined with the notion of locally admissible moves to fully characterise admissible strategies. In the special case of safety objectives, we can show that admissible strategies are exactly those that always play locally admissible moves. This situation is depicted in Figure 2.

Second, we build on our characterisation of admissible strategies based on the notions of locally admissible moves and strongly cooperative optimal strategies to obtain algorithms to solve the *assume admissible synthesis* problem for concurrent games. In the assume admissible synthesis problem, we ask whether a given player has an *admissible* strategy that is *winning* against *all admissible* strategies of the other players. So this rule relaxes the classical synthesis rule by asking for a strategy that is winning against the admissible strategies of the other players only and not against all of them. This is reasonable as in a multi-player game, each player has his own objective which is generally not the complement of the objectives of the other players. The assume-admissible rule makes the hypothesis that players are rational, hence they play admissible strategies and it is sufficient to win against those strategies. Our algorithm is applicable to all ω-regular objectives and it is based on a reduction to a zero-sum two-player game in the sure semantics. While this reduction shares intuitions with the reduction that we proposed in [8] to solve the same problem in the turn-based case, our reduction here is based on games with imperfect information [18]. In contrast, in the turn-based case, games of perfect information are sufficient. The correctness and completeness of our reduction are proved in Theorem 11.

Related works. Concurrent *two player zero-sum* games are studied in [14] and [11]. We rely on the algorithms defined in [11] to compute states from which players have almost surely winning strategies. States where players have (deterministic) winning strategies can be computed by a reduction to more classical turn-based game graphs [2]. Nash equilibria have been studied in concurrent games [6], but without randomised strategies. None of those papers consider the notion of admissibility.

We use the notion of admissibility to obtain synthesis algorithms for systems composed of several sub-systems starting from non zero-sum specifications. Other approaches have been proposed based the notion of Nash equilibria (which suffer from the well-known limitation of non-credible threats): assume-guarantee synthesis [12] and rational synthesis [16, 17]. Those works assume the simpler setting of turn-based games and so they do not deal with randomised strategies.

Finally, in [13], Damm and Finkbeiner use the notion of *dominant strategy* to provide a compositional semi-algorithm for the (undecidable) distributed synthesis problem. However, the notion of dominant strategy is strictly stronger than the notion of admissible strategies, and dominant strategies are not guaranteed to exist, unlike admissible ones.

2 Preliminaries

Concurrent games played on graphs. Let $P = \{1, 2, \dots n\}$ be a set of players. A *concurrent game* played on a finite graph by the players in P is a tuple $\mathcal{G} = (S, \Sigma, s_{\mathsf{init}}, (\Sigma_p)_{p \in P}, \delta)$ where,

(i) S is a finite set of *states*; and $s_{\mathsf{init}} \in S$ the *initial state*;

(ii) Σ is a finite set of *actions*;

(iii) For all $p \in P$, $\Sigma_p : S \to 2^\Sigma \setminus \{\emptyset\}$ is an *action assignment* that assigns, to all states $s \in S$, the set of actions available to player p from state s.

(iv) $\delta : S \times \Sigma \times \ldots \times \Sigma \to S$ is the *transition function.*

We write $\Sigma(s) = \Sigma_1(s) \times \ldots \times \Sigma_n(s)$ for all $s \in S$. It is often convenient to consider a player p separately and see the set of all other players $P \setminus \{p\}$ as a single player denoted $-p$. Hence, the set of actions of $-p$ in state s is: $\Sigma_{-p}(s) =_{\text{def}} \prod_{q \in P \setminus \{p\}} \Sigma_q(s)$. We assume that $\Sigma_i(s) \cap \Sigma_j(s) = \emptyset$ for all $s \in S$ and $i \neq j$. We denote by $\text{Succ}(s, a) = \{\delta(s, a, b) \mid b \in \Sigma_{-p}(s)\}$ the set of possible *successors* of the state $s \in S$ when player p performs action $a \in \Sigma_p(s)$. A particular case of concurrent games are the *turn-based games*. A game $\mathcal{G} = (S, \Sigma, s_{\mathsf{init}}, (\Sigma_p)_{p \in P}, \delta)$ is turn-based iff for all states $s \in S$, there is a unique player p s.t. the successors of s depend only on p's choice of action, i.e., $\text{Succ}(s, a)$ contains exactly one state for all $a \in \Sigma_p(s)$.

A *history* is a finite path $h = s_1 s_2 \ldots s_k \in S^*$ s.t.

(i) $k \in \mathbb{N}$;

(ii) $s_1 = s_{\mathsf{init}}$; and

(iii) for every $2 \leq i \leq k$, there exists $(a_1, \ldots, a_n) \in \Sigma^{|P|}$ with $s_i = \delta(s_{i-1}, a_1, \ldots, a_n)$.

The *length* $|h|$ of a history $h = s_1 s_2 \ldots s_k$ is its number of states k; for every $1 \leq i \leq k$, we denote by h_i the state s_i and by $h_{\leq i}$ the history $s_1 s_2 \ldots s_i$. We denote by $\text{last}(h)$ the last state of h, that is, $\text{last}(h) = h_{|h|}$. A *run* is defined similarly as a history except that its length is infinite. For a run $\rho = s_1 s_2 \ldots \in S^\omega$ and $i \in \mathbb{N}$, we also write $\rho_{\leq i} = s_1 s_2 \ldots s_i$ and $\rho_i = s_i$. Let $\texttt{Hist}(\mathcal{G})$ (resp. $\texttt{Runs}(\mathcal{G})$) denote the set of histories (resp. runs) of $\mathcal{G}$. The game is played from the initial state s_{init} for an infinite number of rounds, producing a run. At each round $i \geq 0$, with current state s_i, all players p select simultaneously an action $a_p^i \in \Sigma_p(s_i)$, and the state $\delta(s_i, a_1^i, \ldots, a_n^i)$ is appended to the current history. The selection of the action by a player is done according to strategies defined below.

Randomised moves and strategies. Given a finite set A, a probability distribution on A is a function $\alpha : A \to [0, 1]$ such that $\sum_{a \in A} \alpha(a) = 1$; and we let $\text{Supp}(\alpha) = \{a \mid \alpha(a) > 0\}$ be the support of α. We denote by $\alpha(B) = \sum_{a \in B} \alpha(a)$ the probability of a given set B according to α. The set of probability distributions on A is denoted by $\mathcal{D}(A)$. A *randomised move* of player p in state s is a probability distribution on $\Sigma_p(s)$, that is, an element of $\mathcal{D}(\Sigma_p(s))$. A randomised move that assigns probability 1 to an action and 0 to the others is called a *Dirac move*. We will henceforth denote randomised moves as sums of actions weighted by their respective probabilities. For instance $0.5f + 0.5g$ denotes the randomised move that assigns probability 0.5 to f and g (and 0 to all other actions). In particular, we denote by b a Dirac move that assigns probability 1 to action b.

Given a state s and a tuple $\boldsymbol{\beta} = (\beta_p)_{p \in P} \in \prod_{p \in P} \mathcal{D}(\Sigma_p(s))$ of randomised moves from s, one per player, we let $\delta_r(s, \boldsymbol{\beta}) \in \mathcal{D}(S)$ be the probability distribution on states s.t. for all $s' \in S$: $\delta_r(s, \boldsymbol{\beta})(s') = \sum_{\boldsymbol{a} \mid \delta(s, \boldsymbol{a}) = s'} \boldsymbol{\beta}(\boldsymbol{a})$, where $\boldsymbol{\beta}(a_1, \ldots, a_n) = \prod_{i=1}^n \beta_i(a_i)$. Intuitively, $\delta_r(s, \boldsymbol{\beta})(s')$ is the probability to reach s' from s when the players play according to $\boldsymbol{\beta}$.

A *strategy* for player p is a function σ from histories to randomised moves (of player p) such that, for all $h \in \texttt{Hist}(\mathcal{G})$: $\sigma(h) \in \mathcal{D}(\Sigma_p(\text{last}(h)))$. A strategy is called Dirac at history h, if $\sigma(h)$ is a Dirac move; it is called Dirac if it is Dirac at all histories. We denote by $\Gamma_p(\mathcal{G})$ the set of player-p strategies in the game, and by $\Gamma_p^{det}(\mathcal{G})$ the set of player-p strategies that only use Dirac moves (those strategies are also called *deterministic*); we might omit $\mathcal{G}$ if it is clear from context. A *strategy profile* $\boldsymbol{\sigma}$ for a subset $A \subseteq P$ of players is a tuple $(\boldsymbol{\sigma}_p)_{p \in A}$ with $\boldsymbol{\sigma}_p \in \Gamma_p$ for all $p \in A$. When the set of players A is omitted, we assume $A = P$. Let $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_p)_{p \in P}$ be a strategy profile. Then, for all players p, we let $\boldsymbol{\sigma}_{-p}$ denote the restriction

of $\boldsymbol{\sigma}$ to $P \setminus \{p\}$ (hence, $\boldsymbol{\sigma}_{-p}$ can be regarded as a strategy of player $-p$ that returns, for all histories h, a randomised move from $\prod_{p \in P \setminus \{p\}} \mathcal{D}(\Sigma_p(s)) \subseteq \mathcal{D}(\Sigma_{-p}(\text{last}(h)))$). We sometimes denote $\boldsymbol{\sigma}$ by the pair $(\boldsymbol{\sigma}_p, \boldsymbol{\sigma}_{-p})$. Given a history h, we let $(\boldsymbol{\sigma}_p)_{p \in A}(h) = (\boldsymbol{\sigma}_p(h))_{p \in A}$.

Let h be a history and let ρ be a history or a run. Then, we write $h \subseteq_{\mathsf{pref}} \rho$ iff h is a prefix of ρ, i.e., $\rho_{\leq |h|} = h$. Given two strategies $\sigma, \sigma' \in \Gamma_p$, and a history h, we let $\sigma\langle h \leftarrow \sigma' \rangle$ be the strategy that follows σ and *shifts* to σ' as soon as h has been played (i.e. $\sigma\langle h \leftarrow \sigma' \rangle$ is s.t. for all histories h': $\sigma\langle h \leftarrow \sigma' \rangle(h') = \sigma'(h')$ if $h \subseteq_{\mathsf{pref}} h'$; and $\sigma\langle h \leftarrow \sigma' \rangle(h') = \sigma(h')$ otherwise).

Probability measure and outcome of a profile. Given a history h, we let $\text{Cyl}(h) = \{\rho \mid h \subseteq_{\mathsf{pref}} \rho\}$ be the *cylinder* of h. To each strategy profile $\boldsymbol{\sigma}$, we associate a probability measure $\mathbb{P}_{\boldsymbol{\sigma}}$ on certain sets of runs. First, for a history h, we define $\mathbb{P}_{\boldsymbol{\sigma}}(\text{Cyl}(h))$ inductively on the length of h: $\mathbb{P}_{\boldsymbol{\sigma}}(\text{Cyl}(s_{\mathsf{init}})) = 1$, and $\mathbb{P}_{\boldsymbol{\sigma}}(\text{Cyl}(h's')) = \mathbb{P}_{\boldsymbol{\sigma}}(\text{Cyl}(h')) \cdot \delta_r(\text{last}(h'), \boldsymbol{\sigma}(h'))(s')$when $|h| > 1$ and $h = h's'$. Based on this definition, we can extend the definition of $\mathbb{P}_{\boldsymbol{\sigma}}$ to any Borel set of runs on cylinders. In particular, the function $\mathbb{P}_{\boldsymbol{\sigma}}$ is well-defined for all ω-regular sets of runs, that we will consider in this paper [19]. We extend the $\texttt{Hist}$ notation and let $\texttt{Hist}(\boldsymbol{\sigma})$ be the set of histories h such that $\mathbb{P}_{\boldsymbol{\sigma}}(\text{Cyl}(h)) > 0$. Given a profile $\boldsymbol{\sigma}$ we denote by $\texttt{Outcome}(\boldsymbol{\sigma})$ the set of runs ρ s.t. all prefixes h of ρ belong to $\texttt{Hist}(\boldsymbol{\sigma})$. In particular, $\mathbb{P}_{\boldsymbol{\sigma}}(\texttt{Outcome}(\boldsymbol{\sigma})) = 1$. Note that when $\boldsymbol{\sigma}$ is composed of Dirac strategies then $\texttt{Outcome}(\boldsymbol{\sigma})$ is a singleton. The outcome (set of histories) of a strategy $\sigma \in \Gamma_p$, denoted by $\texttt{Outcome}(\sigma)$ ($\texttt{Hist}(\sigma)$), is the union of outcomes (set of histories, respectively) of profile $\boldsymbol{\sigma}$ s.t. $\boldsymbol{\sigma}_p = \sigma$.

Winning conditions. To determine the gain of all players in the game $\mathcal{G}$, we define *winning conditions* that can be interpreted with two kinds of semantics denoted by the symbols $\texttt{S}$ for the *sure semantics* or and $\texttt{A}$ for the *almost sure semantics*. A winning condition Φ is a subset of $\texttt{Runs}(\mathcal{G})$ called *winning runs*. From now on, we assume that concurrent games are equipped with a function Φ, called the winning condition, and mapping all players $p \in P$ to a winning condition $\Phi(p)$. A profile $\boldsymbol{\sigma}$ is $\texttt{A}$-winning for $\Phi(p)$ if $\mathbb{P}_{\boldsymbol{\sigma}}(\Phi) = 1$ which we write $\mathcal{G}, \boldsymbol{\sigma} \models^{\texttt{A}} \Phi(p)$. A profile $\boldsymbol{\sigma}$ is $\texttt{S}$-winning for $\Phi(p)$ if $\texttt{Outcome}(\mathcal{G}, \boldsymbol{\sigma}) \subseteq \Phi(p)$ which we write $\mathcal{G}, \boldsymbol{\sigma} \models^{\texttt{S}} \Phi(p)$. Note that when $\boldsymbol{\sigma}$ is Dirac, the two semantics coincide: $\mathcal{G}, \boldsymbol{\sigma} \models^{\texttt{S}} \Phi(p)$ iff $\mathcal{G}, \boldsymbol{\sigma} \models^{\texttt{A}} \Phi(p)$. The profile $\boldsymbol{\sigma}$ is $\texttt{A}$-winning from h if $h \in \texttt{Hist}(\mathcal{G}, \boldsymbol{\sigma})$ and $\mathbb{P}_{\boldsymbol{\sigma}}(\Phi(p) \mid \text{Cyl}(h)) = \mathbb{P}_{\boldsymbol{\sigma}}(\Phi(p) \cap \text{Cyl}(h))/\mathbb{P}_{\boldsymbol{\sigma}}(\text{Cyl}(h)) = 1$ which we denote $\mathcal{G}, \boldsymbol{\sigma} \models^{\texttt{A}}_h \Phi(p)$. The profile $\boldsymbol{\sigma}$ is $\texttt{S}$-winning from h if $\{\rho \in \texttt{Outcome}(\mathcal{G}, \boldsymbol{\sigma}) \mid h \subseteq_{\mathsf{pref}} \rho\} \subseteq \Phi(p)$, which we denote $\mathcal{G}, \boldsymbol{\sigma} \models^{\texttt{S}}_h \Phi(p)$. We often omit $\mathcal{G}$ in notations when clear from the context. Most of our definitions and results hold for both semantics and we often state them using the symbol $\star \in \{\texttt{S}, \texttt{A}\}$ as in the following definition. Given a semantics $\star \in \{\texttt{S}, \texttt{A}\}$, a strategy σ for player p (from a history h) is called $\star$-winning for player p if for every $\tau \in \Gamma_{-p}$, the profile (σ, τ) is $\star$-winning for player p (from h). Note that a strategy σ for player p is $\texttt{S}$-winning iff $\texttt{Outcome}(\sigma) \subseteq \Phi(p)$. We often describe winning conditions using standard linear temporal operators $\Box$ and $\Diamond$; *e.g.* $\Box\Diamond S$ means the set of runs that visit infinitely often S. See [3] for a formal definition.

A winning condition $\Phi(p)$ is *prefix-independent* if for all $s_1 s_2 \ldots \in \Phi(p)$, and all $i \geq 1$: $s_i s_{i+1} \ldots \in \Phi(p)$. When $\Phi(p)$ contains all runs that do not visit some designated set $\mathsf{Bad}_p \subseteq S$ of states, we say that $\Phi(p)$ is a *safety condition*. A safety game is a game whose winning condition Φ is such that $\Phi(p)$ is a safety condition for all players p. Without loss of generality, we assume that safety games are so-called *simple* safety games: a safety game $(S, \Sigma, s_{\mathsf{init}}, (\Sigma_p)_{p \in P}, \delta)$ is *simple* iff for all players p, for all $s \in S$: $s \in \mathsf{Bad}_p$ implies that no $s' \notin \mathsf{Bad}_p$ is reachable from s. That is, once the safety condition is violated, then it remains violated forever at all future histories.

▶ **Example 1.** Let us consider three player-1 strategies in Figure 1.
(i) σ_1 is any strategy that plays a in s_0;
(ii) σ_2 is any strategy that plays b in s_0, d in s_1 and f in s_2; and
(iii) σ_3 is any strategy that plays b in s_0, d in s_1, and $0.5f + 0.5g$ in s_2.
Clearly, σ_1 never allows one to reach Trg while some runs respecting σ_2 and σ_3 do (remember that there is no $\star$-winning strategy in this game). We will see later that the best choice of player 1 (among σ_2, σ_3) depends on the semantics we consider. In the almost-sure semantics, σ_3 is 'better' for player 1, because σ_3 is an $\mathtt{A}$-winning strategy from all histories ending in s_2, while σ_2 is not. On the other hand, in the sure semantics, playing σ_2 is 'better' for player 1 than σ_3. Indeed, for all player-2 strategies τ, either $\mathtt{Outcome}(\sigma_3, \tau)$ contains only runs that do not reach s_2 (hence, do not reach Trg either), or $\mathtt{Outcome}(\sigma_3, \tau)$ contains at least a run that reaches s_2, but, in this case, it also contains a run of the form hs_2^ω that does not reach Trg (because, intuitively, player 1 plays *both* f and g from s_2). So, σ_3 is not $\mathtt{S}$-winning against any τ, while σ_2 wins at least against a player 2 strategy that plays b' in s_0, d' in s_1 and f' in s_2. We formalise these intuitions in the next section.

3 Admissibility

In this section, we define the central notion of the paper: admissibility [5, 9]. Intuitively, a strategy is admissible when it plays 'as well as possible'. Hence the definition of admissible strategies is based on a notion of domination between strategies: a strategy σ' dominates another strategy σ when σ' wins every time σ does. Obviously, players have no interest in playing dominated strategies, hence admissible strategies are those that are not dominated. Apart from these (classical) definitions, we characterise admissible strategies as those that satisfy two weaker notions: they must be both *strongly cooperative optimal* and play only *locally-admissible moves*. Finally, we discuss important characteristics of admissible strategies that will enable us to perform assume-admissible synthesis (see Section 4).

In this section, we fix a game $\mathcal{G}$, a player p, and, following our previous conventions, we denote by Γ_{-p} the set $\{\boldsymbol{\sigma}_{-p} \mid \boldsymbol{\sigma} \in \Gamma\}$.

Admissible strategies. We first recall the classical notion of *admissible strategy* [5, 1]. Given two strategies $\sigma, \sigma' \in \Gamma_p$, we say that σ is $\star$-*weakly dominated* by σ', denoted $\sigma \preccurlyeq^\star \sigma'$, if for all $\tau \in \Gamma_{-p}$: $(\sigma, \tau) \models^\star \Phi(p)$ *implies* $(\sigma', \tau) \models^\star \Phi(p)$. This indeed captures the idea than σ' is *not worse* that σ, because it wins (for p) every time σ does. Note that $\preccurlyeq^\star$ is not anti-symmetric, hence we write $\sigma \approx^\star \sigma'$ when σ and σ' are equivalent, i.e. $\sigma \preccurlyeq^\star \sigma'$ and $\sigma' \preccurlyeq^\star \sigma$. In other words $\sigma \approx^\star \sigma'$ iff for every $\tau \in \Gamma_{-p}$, $(\sigma, \tau) \models^\star \Phi(p) \Leftrightarrow (\sigma', \tau) \models^\star \Phi(p)$. When $\sigma \preccurlyeq^\star \sigma'$ but $\sigma' \not\preccurlyeq^\star \sigma$ we say that σ is $\star$-*dominated* by σ', and we write $\sigma \prec^\star \sigma'$. Observe that $\sigma \prec^\star \sigma'$ holds if and only if $\sigma \preccurlyeq^\star \sigma'$ and there exists at least one $\tau \in \Gamma_{-p}$, such that $(\sigma, \tau) \not\models^\star \Phi(p)$ and $(\sigma', \tau) \models^\star \Phi(p)$. That is, σ' is now *strictly better* than σ. Then, *a strategy σ is* ***$\star$-admissible*** *iff there is no strategy σ' s.t. $\sigma \prec^\star \sigma'$*, i.e., σ is $\star$-admissible iff it is not $\star$-dominated.

▶ **Example 2.** Let us continue our running example, by formalising the intuitions we have sketched in Example 1. Since σ_1 does not allow to reach the target, while some runs respecting σ_2 and σ_3 do, we have: $\sigma_1 \prec^\star \sigma_2$ and $\sigma_1 \prec^\star \sigma_3$. Moreover, $\sigma_2 \prec^{\mathtt{A}} \sigma_3$ because σ_3 is $\mathtt{A}$-winning from any history that ends in s_2 while σ_2 is not because it does not $\mathtt{A}$-win against a player 2 strategy that would always play g' in s_2 (and both strategies behave the same way in s_0 and s_1). On the other hand, $\sigma_3 \prec^{\mathtt{S}} \sigma_2$ since we saw in Example 1 that every profile containing σ_3 is not $\mathtt{S}$-winning while some profiles containing σ_2 are. We will see later that σ_3 is $\mathtt{A}$-admissible and σ_2 is $\mathtt{S}$-admissible.

Values of histories. Before we discuss strongly cooperative optimal and locally admissible strategies, we associate *values* to histories. Let h be a history, and σ be a strategy of player p. Then, the *value of h w.r.t. σ for semantics $\star \in \{\mathtt{S}, \mathtt{A}\}$* is defined as follows. $\chi^\star_\sigma(h) = 1$ if σ is $\star$-winning from h; $\chi^\star_\sigma(h) = 0$ if there are $\tau \in \Gamma_{-p}$ and $\tau' \in \Gamma_{-p}$ s.t. $(\sigma, \tau) \models^\star_h \Phi(p)$, and $(\sigma, \tau') \not\models^\star_h \Phi(p)$; and -1 otherwise.

Value $\chi^\star_\sigma(h) = 1$ corresponds to the case where σ is $\star$-winning for player p from h (thus, against all possible strategies in Γ_{-p}). When $\chi^\star_\sigma(h) = 0$, σ is *not* $\star$-winning from h (because of τ' in the definition), but the other players can still help p to reach his objective (by playing some τ s.t. $(\sigma, \tau) \models^\star_h \Phi(p)$, which exists by definition). Last, $\chi^\star_\sigma(h) = -1$ when there is no hope for p to $\star$-win, even with the collaboration of the other players. In this case, there is no τ s.t. $(\sigma, \tau) \models^\star_h \Phi(p)$. Hence, having $\chi^\star_\sigma(h) = -1$ is stronger than saying that σ is not winning—when σ is not winning, we could have $\chi^\star_\sigma(h) = 0$ as well.

We define the value of a history h for player p as the best value he can achieve with his different strategies: $\chi^\star_p(h) = \max_{\sigma \in \Gamma_p} \chi^\star_\sigma(h)$. Last, for $v \in \{-1, 0, 1\}$, let $\mathtt{Val}^\star_{p,v}$ be the set of histories h s.t. $\chi^\star_p(h) = v$.

Strongly cooperative optimal strategies. We are now ready to define *strongly cooperative optimal* (SCO) strategies. Recall that, in the classical setting of turn-based games, admissible strategies are exactly the SCO strategies [9]. We will see that this condition is still necessary but not sufficient in the concurrent setting.

A strategy σ of Player p is $\star$-SCO at h iff $\chi^\star_\sigma(h) = \chi^\star_p(h)$; and σ is $\star$-SCO iff it is $\star$-SCO at all $h \in \mathtt{Hist}(\sigma)$. Intuitively, when σ is a $\star$-SCO strategy of Player p, the following should hold:

- **(i)** if p has a $\star$-winning strategy from h (i.e. $\chi^\star_p(h) = 1$), then, σ should be $\star$-winning (i.e. $\chi^\star_\sigma(h) = 1$); and
- **(ii)** otherwise if p has no $\star$-winning strategy from h but still has the opportunity to $\star$-win with the help of other players (hence $\chi^\star_p(h) = 0$), then, σ should enable the other players to help p fulfil his objective (i.e. $\chi^\star_\sigma(h) = 0$).

Observe that when $\chi^\star_p(h) = -1$, no continuation of h is $\star$-winning for p, so $\chi^\star_\sigma(h) = -1$ for all strategies σ.

▶ **Example 3.** Consider again the example in Figure 1. For the almost-sure semantics, we have $\mathtt{Val}^{\mathtt{A}}_{p,1} = \{h \mid \mathrm{last}(h) \in \{s_2, \mathsf{Trg}\}\}$, and $\mathtt{Val}^{\mathtt{A}}_{p,0} = \{h \mid \mathrm{last}(h) \in \{s_0, s_1\}\}$. For the sure semantics, we have: $\mathtt{Val}^{\mathtt{S}}_{1,1} = \{h \mid \mathrm{last}(h) = \mathsf{Trg}\}$, and $\mathtt{Val}^{\mathtt{S}}_{1,0} = \{h \mid \mathrm{last}(h) \neq \mathsf{Trg}\}$. Consider again the three strategies σ_1, σ_2 and σ_3 from Example 1. We see that σ_2 is $\mathtt{S}$-SCO but it is not $\mathtt{A}$-SCO because, for all profiles h ending in s_2: $\chi^{\mathtt{A}}_{\sigma_2}(h) = 0$ while $h \in \mathtt{Val}^{\mathtt{A}}_{1,1}$. On the other hand, σ_3 is $\mathtt{A}$-SCO; but it is not $\mathtt{S}$-SCO. Indeed, one can check that, for all strategies $\tau \in \Gamma_2$: if $\mathtt{Outcome}(\sigma_3, \tau)$ contains a run reaching Trg, then it also contains a run that cycles in s_2. So, for all such strategies τ, $\mathtt{Outcome}(\sigma_3, \tau) \not\models^{\mathtt{S}} \Phi(1)$, hence $\chi^{\mathtt{S}}_{\sigma_3}(h) = -1$ for all histories that end in s_2; while $\chi^{\mathtt{S}}_p(h) = 0$ since $\chi^{\mathtt{S}}_{\sigma'}(h) = 0$ for all Dirac strategies σ'.

Next, let us build a strategy σ'_3 that is $\mathtt{A}$-dominated by σ_3 (hence, not $\mathtt{A}$-admissible), but $\mathtt{A}$-SCO. We let σ'_3 play as σ_3 except that σ'_3 plays c the first time s_1 is visited (hence ensuring that the self-loop on s_1 will be taken after the first visit to s_1). Now, σ'_3 is $\mathtt{A}$-dominated by σ_3, because

- **(i)** σ_3 $\mathtt{A}$-wins every time σ'_3 does; but
- **(ii)** σ'_3 does not $\mathtt{A}$-win against the player 2 strategy τ that plays d' only when s_1 is visited for the first time, while σ_3 $\mathtt{A}$-wins against τ.

However, σ'_3 is SCO because playing c keeps the value of the history equal to $0 = \chi^{\mathtt{A}}_1(h)$ (intuitively, playing c once does not prevent the other players from helping in the future).

As similar example can be built in the S semantics. Thus, **there are $\star$-SCO strategies which are not admissible**, so, being $\star$-SCO is not a sufficient criterion for admissibility.

Locally admissible moves and strategies. Let us now discuss another criterion for admissibility, which is more *local* in the sense that it is based on a domination between *moves* available to each player after a given history. Let h be a history, and let α and α' be two randomised moves in $\mathcal{D}(\Sigma_p)$. We say that α is *$\star$-weakly dominated* at h by α' (denoted $\alpha \leqslant^\star_h \alpha'$) iff for all $\sigma \in \Gamma_p$ such that $h \in \mathtt{Hist}(\sigma)$ and $\sigma(h) = \alpha$, there exists $\sigma' \in \Gamma_p$ s.t. $h \in \mathtt{Hist}(\sigma')$, $\sigma'(h) = \alpha'$ and $\sigma \preccurlyeq^\star \sigma'$. Observe that the relation $\leqslant^\star_h$ is not anti-symmetric. We let $\simeq^\star_h$ be the equivalence relation s.t. $\alpha \simeq^\star_h \beta$ iff $\alpha \leqslant^\star_h \beta$ and $\beta \leqslant^\star_h \alpha$. When $\alpha \leqslant^\star_h \alpha'$ but $\alpha' \not\leqslant^\star_h \alpha$ we say that α is *$\star$-dominated* at h by α' and denote this by $\alpha <^\star_h \alpha'$. When a randomised move α is not $\star$-dominated at h, we say that α is *$\star$-locally-admissible* ($\star$-LA) at h. This allows us to define a more local notion of dominated strategy: *a strategy σ of player p is **$\star$-locally-admissible** ($\star$-LA) if $\sigma(h)$ is a $\star$-LA move at h, for all histories h.*

▶ **Example 4.** Consider the Dirac move f and the non-Dirac move $0.5f + 0.5g$ played from s_2 in the example in Figure 1. One can check that $0.5f + 0.5g <^{\mathtt{S}}_{s_2} f$. Indeed, consider a strategy σ s.t. $\sigma(h) = 0.5f + 0.5g$ for some h with $\mathrm{last}(h) = s_2$. Then, playing $\sigma(h)$ from h will never allow Player 1 to reach Trg *surely* at the next step, whatever Player 2 plays; while playing, for instance, f (Dirac move) ensures player 1 to reach Trg surely at the next step, against a Player-2 strategy that plays f'. Thus, σ_2 is S-LA but σ_3 is not.

On the other hand, after every randomised move played in state s_2, the updated state is s_2 or s_3 from which A-winning strategies exist, thus $f \simeq^{\mathtt{A}}_h g \simeq^{\mathtt{A}}_h \lambda f + (1-\lambda)g$ for all $\lambda \in [0,1]$ and all histories h s.t. $\mathrm{last}(h) = s_2$ (so, in particular, $\lambda f + (1-\lambda)g \leqslant^{\mathtt{A}}_h f$ and $\lambda f + (1-\lambda)g \leqslant^{\mathtt{A}}_h g$). It follows that both σ_2 and σ_3 are A-LA. However, in the long run, player 1 needs to play $\lambda f + (1-\lambda)g$, with $\lambda \in (0,1)$, infinitely often in order to A-win. In fact, σ_3 is A-winning from s_2 while σ_2 is not. Thus, **there are $\star$-LA strategies which are not admissible**, so being $\star$-LA is not a sufficient criterion for $\star$-admissibility.

We close this section by several lemmata that allow us to better characterise the notion of LA strategies. First, we observe that, while randomisation might be necessary for winning in certain concurrent games (for example, in Figure 1, no Dirac move allows player 1 to reach Trg *surely* from s_2, while playing repeatedly f and g with equal probability ensures to reach Trg with probability 1) randomisation is useless when a player wants to play only locally admissible moves. This is shown by the next Lemma (point (1)), saying that, *if a randomised move α plays some action a with some positive probability, then α is weakly dominated by the Dirac move a.* However, this does not immediately allow us to characterise admissible moves: some Dirac moves could be dominated (hence non-admissible), and some non-Dirac moves could be admissible too. Points (2) and (3) elucidate this: *among Dirac moves, the non-dominated ones are admissible*, and *a non-Dirac move is admissible iff all the Dirac moves that occur in its support are admissible and equivalent to each other.*

▶ **Lemma 5.** *For all histories h and all randomised moves α:*

(i) *For all $a \in Supp(\alpha)$: $\alpha \leqslant^\star_h a$;*

(ii) *Dirac moves that are not $\star$-dominated at h by another Dirac move are $\star$-LA;*

(iii) *A move α is $\star$-LA at h iff, for all $a \in Supp(\alpha)$:*

1. *a is $\star$-LA at h; and*

2. *$a \simeq^\star_h b$ for all $b \in Supp(\alpha)$.*

▶ **Example 6.** As we have seen in Example 4, $0.5f + 0.5g <^{\mathtt{S}}_{s_2} f$. Note that a strategy σ' s.t. $\sigma'(h) = 0.5f + 0.5g$ for all h with $\mathrm{last}(h) = s_2$ has value $\chi^{\mathtt{S}}_{\sigma'}(h) = -1$, while $\chi^{\mathtt{S}}_1(h) = 0$.

This example seems to suggest that the local dominance of two moves coincide with the natural order on the values of histories that are obtained when playing those moves (in other words $x <_h^\star y$ would hold iff the value of the history obtained by playing x is smaller than or equal to the value obtained by playing y). This is not true for histories of value 0: we have seen that a and b are $\leqslant_h^\star$-incomparable, yet playing a or b from s_0 yields a history with value 0 in all cases (even when s_1 is reached). The next Lemma gives a precise characterisation of the dominance relation between Dirac moves in terms of values:

▶ **Lemma 7.** *For all players p, histories h with $\mathrm{last}(h) = s$ and Dirac moves $a, b \in \Sigma_p(s)$: $a \leqslant_h^\star b$ if, and only if the following conditions hold for every $c \in \Sigma_{-p}(s)$ where we write $s_{(a,c)} = \delta(s,(a,c))$ and $s_{(b,c)} = \delta(s,(b,c))$:*

(i) $\chi_p^\star(hs_{(a,c)}) \leq \chi_p^\star(hs_{(b,c)})$;

(ii) *if* $\chi_p^\star(hs_{(a,c)}) = \chi_p^\star(hs_{(b,c)}) = 0$ *then* $s_{(a,c)} = s_{(b,c)}$.

Characterisation and existence of admissible strategies. Equipped with our previous results, we can now establish the main results of this section. First, we show that *$\star$-admissible strategies are exactly those that are both $\star$-LA and $\star$-SCO* (Theorem 8(i)). Then, we show that *admissible strategies always exist* in concurrent games (Theorem 8(ii)).

▶ **Theorem 8** (Characterisation and existence of admissible strategies)**.** *The following holds for all strategies σ in a concurrent game with semantics $\star \in \{\mathtt{S}, \mathtt{A}\}$:*

(i) *σ is $\star$-admissible iff σ is $\star$-LA and $\star$-SCO; in the special case of simple safety objectives, if σ is $\star$-LA then σ is $\star$-admissible.*

(ii) *there is a $\star$-admissible strategy σ' such that $\sigma \preccurlyeq^\star \sigma'$.*

In particular, point (2) implies that admissible strategies always exist in concurrent games.

▶ **Example 9.** We consider again the example in Figure 1, and consider strategies σ_2 and σ_3 as defined in Example 1. Remember that these two strategies do their best to reach s_2, and that, from s_2, σ_2 plays deterministically f, while σ_3 plays f and g with equal probabilities. From Example 3, we know that σ_2 is S-SCO but not A-SCO; while σ_3 is A-SCO but not S-SCO. Indeed, we have already argued in Example 2 that σ_2 is not A-admissible, and that σ_3 is not S-admissible. However, from Example 4, we know that σ_2 is S-LA and that σ_3 is A-LA. So, by Theorem 8, σ_2 is S-admissible and σ_3 is A-admissible as expected.

Finally, we close the section by a finer characterisation of $\star$-admissible strategies. We show that:

(i) in the sure semantics, there is always an S-admissible strategy that plays Dirac moves only; and

(ii) in the almost-sure semantics, there is always an A-admissible strategy that plays Dirac moves only in histories of values 0 or -1.

The difference between the two semantics should not be surprising, as we know already that randomisation is sometimes needed to win (i.e., from histories of value 1) in the almost sure semantics:

▶ **Proposition 10.** *For all player-p strategies σ in a concurrent game:*

(i) *If σ is S-admissible then there exists a Dirac strategy σ' such that $\sigma \simeq^S \sigma'$.*

(ii) *If σ is A-admissible then there exists a strategy σ' that plays only Dirac moves in histories of value ≤ 0 such that $\sigma \simeq^A \sigma'$.*

4 Assume admissible synthesis

In this section we discuss an *assume-admissible synthesis* framework for concurrent games. With classical synthesis, one tries to compute *winning* strategies for all players, i.e., strategies that *always win* against *all possible strategies* of the other players. Unfortunately, it might be the case that such *unconditionally* winning strategies do not exist, as in our example. As explained in the introduction, the assume-admissible synthesis rule relaxes the classical synthesis rule: instead of searching for strategies that win unconditionally, the new rule requires winning against the *admissible strategies* of the other players. So, a strategy may satisfy the new rule while not winning unconditionally. We claim that winning against admissible strategies is well enough assuming that the players are *rational*; if we assume that players only play strategies that are good for achieving their objectives, *i.e.* admissible ones.

The general idea of the assume-admissible synthesis algorithm is to reduce the problem (in a concurrent n-player game) to the synthesis of a winning strategy *in a 2-player zero-sum concurrent game of imperfect information, in the $\mathcal{S}$-semantics* (even when the original assume-admissible problem is in the $\mathtt{A}$-semantics), where the objective of player 1 is given by an LTL formula. Such games are solvable using techniques presented in [11].

More precisely, from a concurrent game $\mathcal{G}$ in the semantics $\star \in \{\mathtt{S}, \mathtt{A}\}$ and player p, we build a game $\mathcal{G}_p^\star$ with the above characteristics, which is used to decide the assume-admissible synthesis rule. If such a solution exists, our algorithm constructs a witness strategy. For example, the game $\mathcal{G}_1^\star$ corresponding to the game in Figure 1 is given in Figure 3. The main ingredients for this construction are the following.

(i) In $\mathcal{G}_p^\star$, the protagonist is player p, and the second player is $-p$.

(ii) Although randomisation is needed to win in such games in general, we interpret $\mathcal{G}_p^\star$ in the $\mathtt{S}$-semantics only. In fact, we have seen that for the protagonist, Dirac moves suffice in states of value 0; so the only states where he might need randomisation are those of value 1 (randomisation does not matter if the value is -1 since the objective is lost anyway). Hence we define winning condition to be $\Phi(p) \vee \Diamond \mathtt{Val}^\star_{p,1}$ enabling us to consider only histories of values 0 in $\mathcal{G}_p^\star$; and thus hiding the parts of the game where randomisation might be needed. We also prove that we can restrict to Dirac strategies for $-p$ when it comes to admissible strategies.

(iii) In order to restrict the strategies to admissible ones, we only allow $\star$-LA moves in $\mathcal{G}_p^\star$. These moves can be computed by solving classical 2-player games ([2]) using Lemma 7. For example, in Figure 3, moves c and c' are removed since they are not $\mathtt{A}$-LA.

(iv) Last, since $\star$-admissible strategies are those that are both $\star$-LA and $\star$-SCO (see Theorem 8), we also need to ensure that the players play $\star$-SCO. This is more involved than $\star$-LA, as the $\star$-SCO criterion is not *local*, and requires information about the *sequence of actual moves* that have been played, which cannot be deduced, in a concurrent game, from the sequence of visited states. So, we store, in the states of $\mathcal{G}_p^\star$, the moves that have been played by all the players to reach the state. For example, in Figure 3, the state labelled by $s_1, (b, b')$ means that $\mathcal{G}$ has reached s_1, and that the last actions played by the players were b and b' respectively. However, players' strategies must not depend on this extra information since they do not have access to this information in $\mathcal{G}$ either. We thus interpret $\mathcal{G}_p^\star$ as a game of *imperfect information* where all the states labelled by the same state of $\mathcal{G}$ are in the same observation class. We can then encode that the players must play $\star$-SCO strategies in the new objective of the games, which will be given as an LTL formula, as we describe below.

To ensure we can effectively solve subproblems mentioned above, we consider ω-regular objectives. We also restrict ourselves to prefix-independent winning conditions to simplify the presentation. In the case of ω-regular objectives, prefix-independence is not a restrictive hypothesis (we can always compute the product of the game graph with a deterministic parity automaton that accepts the ω-regular objective and consider a parity winning condition). The values of the histories depend thus only on their last states, i.e. for all pairs of histories h_1 and h_2: $\mathrm{last}(h_1) = \mathrm{last}(h_2)$ implies that $\chi^\star_p(h_1) = \chi^\star_p(h_2)$. We denote by $\chi^\star_p(s)$ the value $\chi^\star_p(h)$ of all histories h s.t. $\mathrm{last}(h) = s$. Last, we assume that a player cannot play the same action from two different states, i.e. $\forall s_1 \neq s_2$, $\Sigma_{s_1}(p) \cap \Sigma_{s_2}(p) = \emptyset$. Thus, we say that *move a is $\star$-LA* when a is $\star$-LA from all histories ending in the unique state where a is available.

The game $\mathcal{G}^\star_p$. Let us now describe precisely the construction of $\mathcal{G}^\star_p$. Given an n-player concurrent game $\mathcal{G} = (S, \Sigma, s_{\mathsf{init}}, (\Sigma_p)_{p\in P}, \delta)$ with winning condition Φ considered under semantics $\star \in \{\mathtt{S}, \mathtt{A}\}$, and given a player p, we define the two-player zero-sum concurrent game $\mathcal{G}^\star_p = (\overline{S}, \overline{\Sigma}, \overline{s_{\mathsf{init}}}, (\overline{\Sigma}_p, \overline{\Sigma}_{-p}), \overline{\delta})$ where:

- **(i)** $\overline{S} = S \times \overline{\Sigma}^n \cup \{\overline{s_{\mathsf{init}}}\}$;
- **(ii)** $\overline{\Sigma}$ is the set of Dirac $\star$-LA moves in Σ;
- **(iii)** $\overline{s_{\mathsf{init}}} = s_{\mathsf{init}}$ is the initial state;
- **(iv)** $\overline{\Sigma}_p$ is such that $\overline{\Sigma}_p(s)$ is the set of Dirac $\star$-LA moves of p in s, for all $s \in S$;
- **(v)** $\overline{\Sigma}_{-p}$ is s.t. for all $s \in S$: $\overline{\Sigma}_{-p}(s)$ is the set of moves $\boldsymbol{a}$ of $-p$ in s s.t. for all $q \neq p$, a_q is a Dirac $\star$-LA move;
- **(vi)** $\overline{\delta}$ updates the state according to δ, remembering the last actions played: $\overline{\delta}(\overline{s_{\mathsf{init}}}, \boldsymbol{b}) = (\delta(s_{\mathsf{init}}, \boldsymbol{b}), \boldsymbol{b})$ and $\overline{\delta}((s, \boldsymbol{a}), \boldsymbol{b}) = (\delta(s, \boldsymbol{b}), \boldsymbol{b})$ for all $s \in S$.

Note that the game $\mathcal{G}^\star_p$ depends on whether $\star = \mathtt{A}$ or $\star = \mathtt{S}$ because the two semantics yield different sets of LA-moves. However, we interpret $\mathcal{G}^\star_p$ in the sure semantics, so both players can play Dirac strategies only in $\mathcal{G}^\star_p$.

Let us now explain how we obtain an imperfect information game by defining an *observation function* $\mathfrak{o}$. Note that histories in $\mathcal{G}^\star_p$ are of the form: $\overline{h} = \overline{s_{\mathsf{init}}}(s_1, \boldsymbol{a}_1)(s_2, \boldsymbol{a}_2)\cdots(s_n, \boldsymbol{a}_n)$. Then, let $\mathfrak{o} : \overline{S} \to S$ be the mapping that, intuitively, projects moves away from states. For example, in Figure 3, states with observation s_0 are in the dashed rectangle. That is: $\mathfrak{o}(s, \boldsymbol{a}) = s$ for all states s, and $\mathfrak{o}(\overline{s_{\mathsf{init}}}) = s_{\mathsf{init}}$. We extend $\mathfrak{o}$ to histories recursively: $\mathfrak{o}(\overline{s_{\mathsf{init}}}) = s_{\mathsf{init}}$ and $\mathfrak{o}(h(s_n, \boldsymbol{a}_n)) = \mathfrak{o}(h)s_n$. To make $\mathcal{G}^\star_p$ a game of imperfect information, we request that, in $\mathcal{G}^\star_p$, players play only strategies σ s.t. $\sigma(h_1) = \sigma(h_2)$ whenever $\mathfrak{o}(h_1) = \mathfrak{o}(h_2)$.

We relate the strategies in the original game $\mathcal{G}$ with the strategies in $\mathcal{G}^\star_p$, which we need to extract admissible strategies in $\mathcal{G}$ from the winning strategies in $\mathcal{G}^\star_p$ and thus perform assume-admissible synthesis. First, given a player-p strategy σ in $\mathcal{G}$ (i.e., $\sigma \in \Gamma_p(\mathcal{G})$), we say that a strategy $\overline{\sigma} \in \Gamma^{\mathrm{det}}_p(\mathcal{G}^\star_p)$ is a *realisation* of σ iff:

- **(i)** $\overline{\sigma}$ is Dirac; and
- **(ii)** $\overline{\sigma}(h) \in \mathrm{Supp}(\sigma(h)))$ for all h.

Note that every $\star$-LA strategy $\sigma \in \Gamma_i(\mathcal{G})$ admits realisations σ in $\Gamma_i(\mathcal{G}^\star_p)$. Second, given a player-p Dirac strategy σ in $\mathcal{G}^\star_p$ (i.e., $\sigma \in \Gamma^{\mathrm{det}}_p(\mathcal{G}^\star_p)$) we say that $\hat{\sigma} \in \Gamma_p(\mathcal{G})$ is an *extension* of σ iff, for all $h \in \mathtt{Hist}(\mathcal{G}^\star_p, \sigma)$: $\hat{\sigma}(\mathfrak{o}(h)) = \sigma(h)$.

The assume-admissible synthesis technique. As explained above, the assume-admissible rule boils down to computing a winning strategy $\overline{\sigma}$ for player-p in $\mathcal{G}^\star_p$ w.r.t. the winning condition $\Phi_{\mathcal{G}^\star_p}$, and extracting, from $\overline{\sigma}$, the required admissible strategy in $\mathcal{G}$.

We will now formally define $\Phi_{\mathcal{G}_p^\star}$. Let p be a player (in $\mathcal{G}$); and let us denote by $\mathsf{st}(a)$ the (unique) state from which a is available, for all actions a. We define $\texttt{AfterHelpMove}_p^\star$ as

$$\texttt{AfterHelpMove}_p^\star = \{(s, \boldsymbol{a}) \in \overline{S} \mid \exists s' \in \mathrm{Succ}(\mathsf{st}(a_p), a_p) : \chi_p^\star(s') \geq 0 \wedge s' \neq s \wedge \chi_p^\star(s) = 0\}.$$

That is, when $(s, \boldsymbol{a}) \in \texttt{AfterHelpMove}_p^\star$, in $\mathcal{G}$, player p has played a_p from $\mathsf{st}(a_p)$ and, due to player $-p$'s choice, $\mathcal{G}$ has reached s. However, with another choice of player $-p$, the game could have moved to a different state s' from which $-p$ can help p to win as $\chi_p^\star(s') \geq 0$. Intuitively, in runs that visit states of value 0 infinitely often, states from $\texttt{AfterHelpMove}_p^\star$ should be visited infinitely often for player p to play SCO, i.e. such runs might not be winning, but this cannot be blamed on player p who has sought repeatedly the collaboration of the other players to enforce his objective. Observe further that the definition of this predicate requires the labelling of the states (by actions) we have introduced in $\mathcal{G}_p^\star$. For example, in Figure 3, $\texttt{AfterHelpMove}_2^{\mathtt{A}} = \big\{\big(s_0, (a, b')\big), \big(s_1, (b, b')\big)\big\}$. We let $\Phi_0^\star(p) = \Diamond\neg\texttt{Val}_{p,0}^\star \vee \Phi(p) \vee \Box\Diamond\texttt{AfterHelpMove}_p^\star$ and $\Phi_1^\star(p) = \big(\Diamond\texttt{Val}_{p,1}^\star\big) \to \Phi(p)$. Let us define $\Phi_{\mathcal{G}_p^\star} = \big(\bigwedge_{q\neq p} \Phi_0^\star(q) \wedge \Phi_1^\star(q)\big) \to \big(\Phi(p) \vee \Diamond\texttt{Val}_{p,1}^\star\big)$.

▶ **Theorem 11** (Assume-admissible synthesis)**.** *Player p has a $\star$-admissible strategy σ that is $\star$-winning against all player $-p$ $\star$-admissible strategies in $\mathcal{G}$ iff Player p has an $\mathcal{S}$-winning strategy in $\mathcal{G}_p^\star$ for the objective $\Phi_{\mathcal{G}_p^\star}$. Such a $\star$-admissible strategy σ can be effectively computed (from any player p $\mathcal{S}$-winning strategy in $\mathcal{G}_p^\star$).*

Let us explain how we build a strategy in $\mathcal{G}$ with the desired properties, from any player p strategy enforcing $\Phi_{\mathcal{G}_p^\star}$ in $\mathcal{G}_p^\star$. Remember that $\mathcal{G}_p^\star$ ensures that the players play $\star$-LA moves only. We will use $\Phi_{\mathcal{G}_p^\star}$ to make sure that, when SCO strategies are played by $-p$ (relying on the extra information we have encoded in the states), then p reaches a state of value 1. First, consider $\Phi_0^\star(q)$ for $q \neq p$. Runs that satisfy this formula are either those that visit states of value 0 only finitely often ($\Diamond\neg\texttt{Val}_{q,0}^\star$); or those that stay in states of value 0, in which case they must be either winning ($\Phi(q)$) or visit infinitely often states where Player q could have been helped by the other players ($\Box\Diamond\texttt{AfterHelpMove}_q^\star$). This is a necessary condition on runs visiting only value 0 states for the strategy to be SCO. Next, observe that $\Phi_1^\star(q)$ states that *if* a history of value 1 is entered *then* Player q must win. This allows us to understand the left part of the implication in $\Phi_{\mathcal{G}_p^\star}$: the implication can be read as 'if all other players play a $\star$-admissible strategy, then either p should win ($\Phi(p)$) or a state of value 1 for player p should eventually be visited ($\Diamond\texttt{Val}_{p,1}^\star$)'. Then a strategy $\hat{\sigma}$ (in $\mathcal{G}$) that wins against admissible strategies can be extracted from a winning strategy $\overline{\sigma}$ (in $\mathcal{G}_p^\star$) in a straightforward way, *except* when $\overline{\sigma}$ enforces to reach a state of value 1 ($\Diamond\texttt{Val}_{p,1}^\star$ in $\Phi_{\mathcal{G}_p^\star}$). In this case, σ cannot follow $\overline{\sigma}$, but must rather switch to a *winning* strategy, which:

(i) is guaranteed to exist since the state that has been reached has value 1; and

(ii) can be computed using classical techniques [11].

The strategy $\hat{\sigma}$ is not necessarily admissible but by Theorem 8 (1), there is an admissible strategy σ with $\hat{\sigma} \preccurlyeq^\star \sigma$. By weak domination, σ wins against more profiles than $\hat{\sigma}$, in particular, against the profiles of admissible strategies of the other players.

▶ **Example 12.** In our running example, observe that $\neg\texttt{Val}_{2,0}^{\mathtt{A}} = \texttt{Val}_{2,1}^{\mathtt{A}} = \{Win\}$ since there is no state of value -1 in $\mathcal{G}$. Hence, $\Phi(2) = \Diamond Win = \Diamond\texttt{Val}_{2,1}^{\mathtt{A}} = \Diamond\neg\texttt{Val}_{2,0}^{\mathtt{A}}$. Finally, $\texttt{AfterHelpMove}_2^{\mathtt{A}} = \big\{\big(s_0, (a, b')\big), \big(s_1, (b, b')\big)\big\}$, so, after simplification: $\Phi_{\mathcal{G}_1^{\mathtt{A}}} = \big[\Diamond Win \vee \Box\Diamond\big((s_0, (a, b')) \vee (s_1, (b, b'))\big)\big] \to \Diamond Win$. Thus, to win in $\mathcal{G}_1^{\mathtt{A}}$ (under the *sure* semantics), player 1 must ensure to reach Win as long as player 2 visits the set of *bold* states in Figure 3 infinitely often. A winning strategy $\overline{\sigma}$ in $\mathcal{G}_1^{\mathtt{A}}$ consists in (eventually) always playing b from

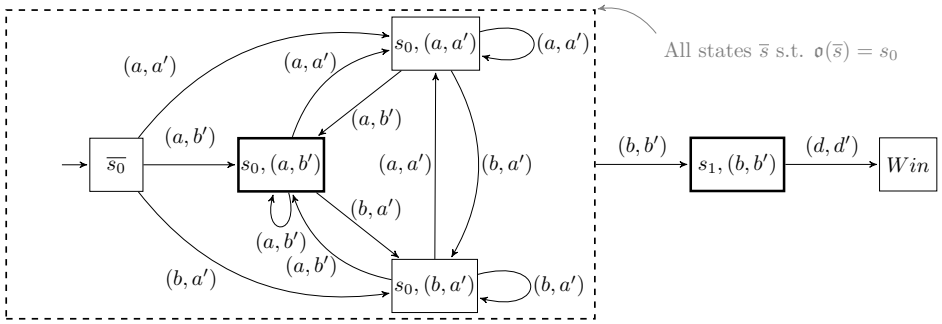


Figure 3 The game $\mathcal{G}_1^{\mathtt{A}}$ obtained from the game in Figure 1. Bold states $\big(s_0, (a, b')\big)$ and $\big(s_1, (b, b')\big)$ are the states of $\mathtt{AfterHelpMove}_2^{\mathtt{A}}$. There is a (b, b')-labelled transition from all states in the dashed rectangle to $\big(s_1, (b, b')\big)$.

all states in the dashed rectangle; and d from $\big(s_1, (b, b')\big)$. Observe that this strategy is compatible with $\mathfrak{o}$. From $\overline{\sigma}$, we can extract an admissible player 1 strategy in $\mathcal{G}$: always play b in s_0; always play d in s_1; and play a winning strategy from s_2 (which is of value 1), for instance: always play $0.5f + 0.5g$ from s_2 like σ_3 does.

We conclude by two remarks on simple safety games and on the choice of our semantics. First, note that assume-admissible synthesis is simpler in simple safety games, since the admissible strategies are exactly the $\star$-LA strategies in this case (see Theorem 8). So, one can build $\mathcal{G}_p$ from $\mathcal{G}$ by pruning the actions which are not $\star$-LA (the labelling by actions is not necessary anymore), and look for a player p winning strategy. Second, in the semantics of concurrent games considered in this paper, players see, at each step, the transition taken but not the actual moves of the other player even once they are played. An alternative semantics could be that the players discover simultaneously the moves of other players after each step, as in the Rock-Paper-Scissors game. The former semantics is more general than the latter since moves played at the preceding round can always be encoded in the current state (as we did in the construction of $\mathcal{G}_p^\star$). Our results remain meaningful in this simpler case (in particular the characterisation of admissible strategies), but assume-admissible synthesis can be performed by reducing to games with *perfect* information.

References

1 Brandenburger Adam, Friedenberg Amanda, H Jerome, et al. Admissibility in games. *Econometrica*, 2008.

2 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002. doi:10.1145/585265.585270.

3 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

4 Nicolas Basset, Gilles Geeraerts, Jean-François Raskin, and Ocan Sankur. Admissibility in concurrent games. *CoRR*, abs/1702.06439, 2017. URL: http://arxiv.org/abs/1702.06439.

5 Dietmar Berwanger. Admissibility in infinite games. In *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings*, number 4393 in Lecture Notes in Computer Science, pages 188–199. Springer, 2007. doi:10.1007/978-3-540-70918-3_17.

6 Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Nash equilibria in concurrent games with Büchi objectives. In *Proceedings of the 31st Conference on Found-*

ations of Software Technology and Theoretical Computer Science (FSTTCS'11), volume 13 of *Leibniz International Proceedings in Informatics*, pages 375–386, Mumbai, India, dec 2011. Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2011.375.

7 Romain Brenguier, Guillermo A. Pérez, Jean-François Raskin, and Ocan Sankur. Admissibility in Quantitative Graph Games. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016)*, volume 65 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FSTTCS.2016.42.

8 Romain Brenguier, Jean-François Raskin, and Ocan Sankur. Assume-admissible synthesis. In Luca Aceto and David de Frutos-Escrig, editors, *CONCUR*, volume 42 of *LIPIcs*, pages 100–113. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.100.

9 Romain Brenguier, Jean-François Raskin, and Ocan Sankur. Assume-admissible synthesis. *Acta Inf.*, 54(1):41–83, 2017. doi:10.1007/s00236-016-0273-2.

10 Romain Brenguier, Jean-François Raskin, and Mathieu Sassolas. The complexity of admissibility in omega-regular games. In *CSL-LICS '14, 2014*. ACM, 2014. doi:10.1145/2603088.2603143.

11 Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Qualitative concurrent parity games. *ACM Trans. Comput. Log.*, 12(4):28:1–28:51, 2011. doi:10.1145/1970398.1970404.

12 Krishnendu Chatterjee and Thomas A. Henzinger. Assume-guarantee synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings*, volume 4424 of *Lecture Notes in Computer Science*, pages 261–275. Springer, 2007.

13 Werner Damm and Bernd Finkbeiner. Automatic compositional synthesis of distributed systems. In *FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings*, volume 8442 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 2014.

14 Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theor. Comput. Sci.*, 386(3):188–217, 2007. doi:10.1016/j.tcs.2007.07.008.

15 Marco Faella. Admissible strategies in infinite games over graphs. In *MFCS 2009*, volume 5734 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 2009.

16 Dana Fisman, Orna Kupferman, and Yoad Lustig. Rational synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 190–204. Springer, 2010.

17 Orna Kupferman, Giuseppe Perelli, and Moshe Y. Vardi. Synthesis with rational environments. In *Multi-Agent Systems - 12th European Conference, EUMAS 2014, Prague, Czech Republic, December 18-19, 2014, Revised Selected Papers*, pages 219–235. Springer, 2014.

18 Jean-François Raskin, Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Algorithms for omega-regular games with imperfect information. *Logical Methods in Computer Science*, 3(3), 2007. doi:10.2168/LMCS-3(3:4)2007.

19 Moshe Y Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 327–338. IEEE, 1985.

Improved Algorithms for Computing the Cycle of Minimum Cost-to-Time Ratio in Directed Graphs*†

Karl Bringmann[1], Thomas Dueholm Hansen[2], and Sebastian Krinninger[3]

1 **Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany**
kbringma@mpi-inf.mpg.de
2 **Aarhus University, Aarhus, Denmark**
tdh@cs.au.dk
3 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
sebastian.krinninger@univie.ac.at

Abstract

We study the problem of finding the cycle of minimum cost-to-time ratio in a directed graph with n nodes and m edges. This problem has a long history in combinatorial optimization and has recently seen interesting applications in the context of quantitative verification. We focus on strongly polynomial algorithms to cover the use-case where the weights are relatively large compared to the size of the graph. Our main result is an algorithm with running time $\tilde{O}(m^{3/4}n^{3/2})$, which gives the first improvement over Megiddo's $\tilde{O}(n^3)$ algorithm [JACM'83] for sparse graphs.[1] We further demonstrate how to obtain both an algorithm with running time $n^3/2^{\Omega(\sqrt{\log n})}$ on general graphs and an algorithm with running time $\tilde{O}(n)$ on constant treewidth graphs. To obtain our main result, we develop a parallel algorithm for negative cycle detection and single-source shortest paths that might be of independent interest.

1998 ACM Subject Classification F.2.2 Computations on discrete structures, G.2.2 Graph algorithms

Keywords and phrases quantitative verification and synthesis, parametric search, shortest paths, negative cycle detection

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.124

1 Introduction

We revisit the problem of computing the cycle of minimum cost-to-time ratio (short: minimum ratio cycle) of a directed graph in which every edge has a cost and a transit time. The problem has a long history in combinatorial optimization and has recently become relevant to the computer-aided verification community in the context of quantitative verification and synthesis of reactive systems [11, 13, 24, 7, 10, 6, 14]. The shift from quantitative to qualitative properties is motivated by the necessity of taking into account the resource

* Full version of this paper available at https://arxiv.org/abs/1704.08122.

† The work of K. Bringmann was partially done while visiting Aarhus University. T. D. Hansen was supported by the Carlsberg Foundation, grant no. CF14-0617. The work of S. Krinninger was partially done while visiting Aarhus University and while at Max Planck Institute for Informatics.

[1] We use the notation $\tilde{O}(\cdot)$ to hide factors that are polylogarithmic in n.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 124; pp. 124:1–124:16

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

consumption of systems (such as embedded systems) and not just their correctness. For algorithmic purposes, these systems are usually modeled as directed graphs where vertices correspond to states of the system and edges correspond to transitions between states. Weights on the edges model the resource consumption of transitions. In our case, we allow two types of resources (called cost and transit time) and are interested in optimizing the ratio between the two quantities. By giving improved algorithms for finding the minimum ratio cycle we contribute to the algorithmic progress that is needed to make the ideas of quantitative verification and synthesis applicable.

From a purely theoretic point of view, the minimum ratio problem is an interesting generalization of the minimum mean cycle problem.[2] A natural question is whether the running time for the more general problem can match the running time of computing the minimum cycle mean (modulo lower order terms). In terms of weakly polynomial algorithms, the answer to this question is yes, since a binary search over all possible values reduces the problem to negative cycle detection. In terms of strongly polynomial algorithms, with running time independent of the encoding size of the edge weights, the fastest algorithm for the minimum ratio cycle problem is due to Megiddo [42] and runs in time $\tilde{O}(n^3)$, whereas the minimum mean cycle can be computed in $O(mn)$ time with Karp's algorithm [38]. This has left an undesirable gap in the case of sparse graphs for more than three decades.

Our results. We improve upon this situation by giving a strongly polynomial time algorithm for computing the minimum ratio cycle in time $O(m^{3/4}n^{3/2}\log^2 n)$ (Theorem 21 in Section 4). We obtain this result by designing a suitable parallel negative cycle detection algorithm and combining it with Megiddo's parametric search technique [42]. We first present a slightly simpler randomized version of our algorithm with one-sided error and the same running time (Theorem 18 in Section 3).

As a side result, we develop a new parallel algorithm for negative cycle detection and single-source shortest paths (SSSP) that we use as a subroutine in the minimum ratio cycle algorithm. This new algorithm has work $\tilde{O}(mn + n^3h^{-3})$ and depth $O(h)$ for any $\log n \leq h \leq n$. Our algorithm uses techniques from the parallel transitive closure algorithm of Ullman and Yannakakis [52] (in particular as reviewed in [39]) and our contribution lies in extending these techniques to directed graphs with positive *and negative* edge weights. In particular, we partially answer an open question by Shi and Spencer [50] who previously gave similar trade-offs for single-source shortest paths in *undirected* graphs with positive edge weights. We further demonstrate how the parametric search technique can be applied to obtain minimum ratio cycle algorithms with running time $\tilde{O}(n)$ on constant treewidth graphs (Corollary 24 in Section 5). Our algorithms do not use fast matrix multiplication. We finally show that if fast matrix multiplication is allowed then slight further improvements are possible, specifically we present an $n^3/2^{\Omega(\sqrt{\log n})}$ time algorithm on general graphs (see full version of this paper).

Prior Work. The minimum ratio problem was introduced to combinatorial optimization in the 1960s by Dantzig, Blattner, and Rao [22] and Lawler [40]. The existing algorithms can be classified according to their running time bounds as follows: strongly polynomial algorithms, weakly polynomial algorithms, and pseudopolynomial algorithms. In terms of strongly polynomial algorithms for finding the minimum ratio cycle we are aware of the following two results:

[2] In the minimum cycle mean problem we assume the transit time of each edge is 1.

- $O(n^3 \log n + mn \log^2 n)$ time using Megiddo's second algorithm [42] together with Cole's technique to reduce a factor of $\log \log n$ [21],
- $O(mn^2)$ time using Burn's primal-dual algorithm [9].

For the class of weakly polynomial algorithms, the best algorithm is to follow Lawler's binary search approach [40, 41], which solves the problem by performing $O(\log(nW))$ calls to a negative cycle detection algorithm. Here $W = O(CT)$ if the costs are given as integers from 1 to C and the transit times are given as integers from 1 to T. Using an idea for efficient search of rationals [47], a somewhat more refined analysis by Chatterjee et al. [14] reveals that it suffices to call the negative cycle detection algorithm $O(\log(|a \cdot b|))$ times when the value of the minimum ratio cycle is $\frac{a}{b}$. Since the initial publication of Lawler's idea, the state of the art in negative cycle detection algorithms has become more diverse. Each of the following five algorithms gives the best known running time for some range of parameters (and the running times have to be multiplied by the factor $\log(nW)$ or $\log(|a \cdot b|)$ to obtain an algorithm for the minimum ratio problem):

- $O(mn)$ time using a variant of the Bellman-Ford algorithm [26, 3, 45],
- $n^3/2^{\Omega(\sqrt{\log n})}$ time using a recent all-pairs shortest paths (APSP) algorithm by Williams [53, 12],
- $\tilde{O}(n^\omega W)$ time using fast matrix multiplication [48, 54], where $2 \leq \omega < 2.3728639$ is the matrix multiplication coefficient [29],
- $O(m\sqrt{n} \log W)$ time using Goldberg's scaling algorithm [31],
- $\tilde{O}(m^{10/7} \log W)$ time using the interior point method based algorithm of Cohen et al. [20]

The third group of minimum ratio cycle algorithms has a pseudopolynomial running time bound. After some initial progress [34, 30, 35], the state of the art is an algorithm by Hartmann and Orlin [33] that has a running time of $O(mnT)$.[3] Other algorithmic approaches, without claiming any running time bounds superior to those reported above, were given by Fox [27], v. Golitschek [32], and Dasdan, Irani, and Gupta [23].

Recently, the minimum ratio problem has been studied specifically for the special case of constant treewidth graphs by Chatterjee, Ibsen-Jensen, and Pavlogiannis [14]. The state of the art for negative cycle detection on constant treewidth graphs is an algorithm by Chaudhuri and Zaroliagis with running time $O(n)$ [17], which by Lawler's binary search approach implies an algorithm for the minimum ratio problem with running time $O(n \log(nW))$. Chatterjee et al. [14] report a running time of $O(n \log(|a \cdot b|))$ based on the more refined binary search mentioned above and additionally give an algorithm that uses $O(\log n)$ space (and hence polynomial time).

As a subroutine in our minimum ratio cycle algorithm, we use a new parallel algorithm for negative cycle detection and single-source shortest paths. The parallel SSSP problem has received considerable attention in the literature [50, 39, 18, 8, 49, 19, 43, 44, 5], but we are not aware of any parallel SSSP algorithm that works in the presence of negative edge weights (and thus solves the negative cycle detection problem). To the best of our knowledge, the only strongly polynomial bounds reported in the literature are as follows: For weighted, directed graphs with non-negative edge weights, Broda, Träff, and Zaroliagis [8] give an implementation of Dijkstra's algorithm with $O(m \log n)$ work and $O(n)$ depth. For weighted, undirected graphs with positive edge weights, Shi and Spencer [49] gave (1) an algorithm with $O(n^3 t^{-2} \log n \log(nt^{-1}) + m \log n)$ work and $O(t \log n)$ depth and (2) an algorithm with $O((n^3 t^{-3} + mnt^{-1}) \log n)$ work and $O(t \log n)$ depth, for any $\log n \leq t \leq n$.

[3] Note that the more fine-grained analysis of Hartmann and Orlin actually gives a running time of $O(m(\sum_{u \in V} \max_{e=(u,v)} t(e)))$.

2 Preliminaries

In the following, we review some of the tools that we use in designing our algorithm.

2.1 Parametric Search

We first explain the parametric search technique as outlined in [1]. Assume we are given a property $\mathcal{P}$ of real numbers that is *monotone* in the following way: if $\mathcal{P}(\lambda)$ holds, then also $\mathcal{P}(\lambda')$ holds for all $\lambda' < \lambda$. Our goal is to find λ^*, the *maximum* λ such that $\mathcal{P}(\lambda)$ holds. In this paper for example, we will associate with each λ a weighted graph G_λ and $\mathcal{P}$ is the property that G_λ has no negative cycle. Assume further that we are given an algorithm $\mathcal{A}$ for deciding, for a given λ, whether $\mathcal{P}(\lambda)$ holds. If λ were known to only assume integer or rational values, we could solve this problem by performing binary search with $O(\log W)$ calls to the decision algorithm, where W is the number of possible values for λ. However, this solution has the drawback of not yielding a strongly polynomial algorithm.

In parametric search we run the decision algorithm 'generically' at the maximum λ^*. As the algorithm does not know λ^*, we need to take care of its control flow ourselves and any time the algorithm performs a comparison we have to 'manually' evaluate the comparison on behalf of the algorithm. If each comparison takes the form of testing the sign of an associated low-degree polynomial $p(\lambda)$, this can be done as follows. We first determine all roots of $p(\lambda)$ and check if $\mathcal{P}(\lambda')$ holds for each such root λ' using another instance of the decision algorithm $\mathcal{A}$. This gives us an interval between successive roots containing λ^* and we can thus resolve the comparison. With every comparison we make, the interval containing λ^* shrinks and at the end of this process we can output a single candidate. If the decision algorithm $\mathcal{A}$ has a running time of T, then the overall algorithm for computing λ^* has a running time of $O(T^2)$.

A more sophisticated use of the technique is possible, if in addition to a sequential decision algorithm $\mathcal{A}_\mathrm{s}$ we have an efficient parallel decision algorithm $\mathcal{A}_\mathrm{p}$. The parallel algorithm performs its computations simultaneously on P_p processors. The number of parallel computation steps until the last processor is finished is called the *depth* D_p of the algorithm, and the number of operations performed by all processors in total is called the *work* W_p of the algorithm.[4] For parametric search, we actually only need parallelism w.r.t. comparisons involving the input values. We denote by the *comparison depth* of $\mathcal{A}_\mathrm{p}$ the number of parallel comparisons (involving input values) until the last processor is finished.

We proceed similar to before: We run $\mathcal{A}_\mathrm{p}$ 'generically' at the maximum λ^* and (conceptually) distribute the work among P_p processors. Now in each parallel step, we might have to resolve up to P_p comparisons. We first determine all roots of the polynomials associated to these comparisons. We then perform a binary search among these roots to determine the interval of successive roots containing λ^* and repeat this process of resolving comparisons at every parallel step to eventually find out the value of λ^*. If the sequential decision algorithm $\mathcal{A}_\mathrm{s}$ has a running time of T_s and the parallel decision algorithm runs on P_p processors in D_p parallel steps, then the overall algorithm for computing λ^* has a running time of $O(P_\mathrm{p}D_\mathrm{p} + D_\mathrm{p}T_\mathrm{s}\log P_\mathrm{p})$. Formally, the guarantees of the technique we just described can be summarized as follows.

[4] To be precise, we use an abstract model of parallel computation as formalized in [28] to avoid distraction by details such as read or write collisions typical to PRAM models.

▶ **Theorem 1** ([1, 42]). *Let $\mathcal{P}$ be a property of real numbers such that if $\mathcal{P}(\lambda)$ holds, then also $\mathcal{P}(\lambda')$ holds for all $\lambda' < \lambda$ and let $\mathcal{A}_p$ and $\mathcal{A}_s$ be algorithms deciding for a given λ whether $\mathcal{P}(\lambda)$ holds such that*

- *the control flow of $\mathcal{A}_p$ is only governed by comparisons that test the sign of an associated polynomial in λ of constant degree,*
- *$\mathcal{A}_p$ is a parallel algorithm with work W_p and comparison depth D_p, and*
- *$\mathcal{A}_s$ is a sequential algorithm with running time T_s.*

Then there is a (sequential) algorithm for finding the maximum value λ such that $\mathcal{P}(\lambda)$ holds with running time $O(W_p + D_p T_s \log W_p)$.

Note that $\mathcal{A}_\mathrm{p}$ and $\mathcal{A}_\mathrm{s}$ need not necessarily be different algorithms. In most cases however, the fastest sequential algorithm might be the better choice for minimizing running time.

2.2 Characterization of Minimum Ratio Cycle

We consider a directed graph $G = (V, E, c, t)$, in which every edge $e = (u, v)$ has a cost $c(e)$ and a transit time $t(e)$. We want to find the cycle C that minimizes the cost-to-time ratio $\sum_{e \in C} c(e) / \sum_{e \in C} t(e)$.

For any real λ define the graph $G_\lambda = (V, E, w_\lambda)$ as the modification of G with weight $w_\lambda(e) = c(e) - \lambda t(e)$ for every edge $e \in E$. The following structural lemma is the foundation of many algorithmic approaches towards the problem.

▶ **Lemma 2** ([22, 41]). *Let λ^* be the value of the minimum ratio cycle of G.*

- *For $\lambda > \lambda^*$, the value of the minimum weight cycle in G_λ is < 0.*
- *The value of the the minimum weight cycle in G_{λ^*} is 0. Each minimum weight cycle in G_{λ^*} is a minimum ratio cycle in G and vice versa.*
- *For $\lambda < \lambda^*$, the value of the minimum weight cycle in G_λ is > 0.*

The obvious algorithmic idea now is to find the right value of λ with a suitable search strategy and reduce the problem to a series of negative cycle detection instances.

2.3 Characterization of Negative Cycle

▶ **Definition 3.** A *potential function* $p\colon V \to \mathbb{R}$ assigns a value to each vertex of a weighted directed graph $G = (V, E, w)$. We call a potential function p *valid* if for every edge $e = (u, v) \in E$, the condition $p(u) + w(e) \geq p(v)$ holds.

The following two lemmas outline an approach for negative cycle detection.

▶ **Lemma 4** ([25]). *A weighted directed graph contains a negative cycle if and only if it has no valid potential function.*

▶ **Lemma 5** ([37]). *Let $G = (V, E, w)$ be a weighted directed graph and let $G' = (V', E', w')$ be the supergraph of G consisting of the vertices $V' = V \cup \{s'\}$ (i.e. with an additional super-source s'), the edges $E' = E \cup \{s'\} \times V$ and the weight function w' given by $w'(s', v) = 0$ for every vertex $v \in V$ and $w'(u, v) = w(u, v)$ for all pairs of vertices $u, v \in V$. If G does not contain a negative cycle, then the potential function p defined by $p(v) = d_{G'}(s', v)$ for every vertex $v \in V$ is valid for G.*

Thus, an obvious strategy for negative cycle detection is to design a single-source shortest paths algorithm that is correct whenever the graph contains no negative cycle. If the graph contains no negative cycle, then the distances computed by the algorithm can be verified to

be a valid potential. If the graph does contain a negative cycle, then the distances computed by the algorithm will not be a valid potential (because a valid potential does not exist) and we can verify that the potential is not valid.

2.4 Computing Shortest Paths in Parallel

In our algorithm we use two building blocks for computing shortest paths in the presence of negative edge weights in parallel. The first such building block was also used by Megiddo [42].

▶ **Observation 6.** *By repeated squaring of the min-plus matrix product, all-pairs shortest paths in a directed graph with real edge weights can be computed using work $O(n^3 \log n)$ and depth $O(\log n)$.*

The second building block is a subroutine for computing the following restricted version of shortest paths.

▶ **Definition 7.** The *shortest h-hop path* from a vertex s to a vertex t is the path of minimum weight among all paths from s to t with at most h edges.

Note that a shortest h-hop path from s to t does not exist, if all paths from s to t use more than h edges. Furthermore, if there is a shortest path from s to t with at most h edges, then the h-hop shortest path from s to t is a shortest path as well. Shortest h-hop paths can be computed by running h iterations of the Bellman-Ford algorithm [26, 3, 45].[5] Similar to shortest paths, shortest h-hop paths need not be unique. We can enforce uniqueness by putting some arbitrary but fixed order on the vertices of the graph and sorting paths according to the induced lexicographic order on the sequence of vertices of the paths. Note that the Bellman-Ford algorithm can easily be adapted to optimizing lexicographically as its second objective.

▶ **Observation 8.** *By performing h iterations of the Bellman-Ford algorithm, the lexicographically smallest shortest h-hop path from a designated source vertex s to each other vertex in a directed graph with real edge weights can be computed using work $O(mh)$ and depth $O(h)$.*

We denote by $\pi(s,t)$ the lexicographically smallest shortest path from s to t and by $\pi^h(s,t)$ the lexicographically smallest shortest h-hop path from s to t. We denote by $V(\pi^h(s,t))$ and $E(\pi^h(s,t))$ the set of nodes and edges of $\pi^h(s,t)$, respectively.

2.5 Approximate Hitting Sets

▶ **Definition 9.** Given a collection of sets $\mathcal{S} \subseteq 2^U$ over a universe U, a hitting set is a set $T \subseteq H$ that has non-empty intersection with every set of $\mathcal{S}$ (i.e., $S \cap T \neq \emptyset$ for every $S \in \mathcal{S}$).

Computing a hitting set of minimum size is an NP-hard problem. For our purpose however, rough approximations are good enough. The first method to get a sufficiently small hitting set uses a simple randomized sampling idea and was introduced to the design of graph algorithms by Ullman and Yannakakis [52]. We use the following formulation.

▶ **Lemma 10.** *Let $c \geq 1$, let U be a set of size s and let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a collection of sets over the universe U of size at least q. Let T be a subset of U that was obtained by choosing*

[5] The first explicit use of the Bellman-Ford algorithm to compute shortest h-hop paths that we are aware of is in Thorup's dynamic APSP algorithm [51].

each element of U independently with probability $p = \min(x/q, 1)$ where $x = c \ln(ks) + 1$. Then, with high probability (whp), i.e., probability at least $1 - 1/s^c$, the following two properties hold:

1. *For every $1 \leq i \leq k$, the set S_i contains an element of T, i.e., $S_i \cap T \neq \emptyset$.*
2. *$|T| \leq 3xs/q = O(cs \log(ks)/q)$.*

The second method is to use a heuristic to compute an approximately minimum hitting set. In the sequential model, a simple greedy algorithm computes an $O(\log n)$-approximation [36, 2]. We use the following formulation.

▶ **Lemma 11.** *Let U be a set of size s and let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a collection of sets over the universe U of size at least q. Consider the simple greedy algorithm that picks an element u in U that is contained in the largest number of sets in $\mathcal{S}$ and then removes all sets containing u from $\mathcal{S}$, repeating this step until $\mathcal{S} = \emptyset$. Then the set T of elements picked by this algorithm satisfies:*

1. *For every $1 \leq i \leq k$, the set S_i contains an element of T, i.e., $S_i \cap T \neq \emptyset$.*
2. *$|T| \leq O(s \log(k)/q)$.*

Proof. We follow the standard proof of the approximation ratio $O(\log n)$ for the greedy set cover heuristic. The first statement is immediate, since we only remove sets when they are hit by the picked element. Since each of the k sets contains at least q elements, on average each element in U is contained in at least kq/s sets. Thus, the element u picked by the greedy algorithm is contained in at least kq/s sets. The remaining number of sets is thus at most $k - kq/s = k(1 - q/s)$. Note that the remaining sets still have size at least q, since they do not contain the picked element u. Inductively, we thus obtain that after i iterations the number of remaining sets is at most $k(1 - q/s)^i$, so after $O(\log(k) \cdot s/q)$ iterations the number of remaining sets is less than 1 and the process stops. ◀

The above greedy algorithm is however inherently sequential and thus researchers have studied more sophisticated algorithms for the parallel model. The state of the art in terms of deterministic algorithms is an algorithm by Berger et al. [4][6].

▶ **Theorem 12** ([4]). *Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a collection of sets over the universe U, let $n = |U|$ and $m = \sum_{1 \leq i \leq k} |S_i|$. For $0 < \epsilon < 1$, there is an algorithm with work $O((m+n)\epsilon^{-6} \log^4 n \log m \log^6(nm))$ and depth $O(\epsilon^{-6} \log^4 n \log m \log^6(nm))$ that produces a hitting set of $\mathcal{S}$ of size at most $(1+\epsilon)(1+\ln \Delta) \cdot OPT$, where Δ is the maximum number of occurrences of any element of U in $\mathcal{S}$ and OPT is the size of a minimum hitting set.*

3 Randomized Algorithm for General Graphs

3.1 A Parallel SSSP Algorithm

In the following we design a parallel SSSP algorithm that can be used to check for negative cycles. Formally, we will in this subsection prove the following statement.

[6] Berger et al. actually give an approximation algorithm for the following slightly more general problem: Given a hypergraph $H = (V, E)$ and a cost function $c\colon V \to \mathbb{R}$ on the vertices, find a minimum cost subset $R \subseteq V$ that covers H, i.e., an R that minimizes $c(R) = \sum_{v \in R} c(v)$ subject to the constraint $e \cap R \neq \emptyset$ for all $e \in E$.

▶ **Theorem 13.** *There is an algorithm that, given a weighted directed graph $G = (V, E, w)$ containing no negative cycles, computes the shortest paths from a designated source vertex s to all other vertices spending $O(mn \log n + n^3 h^{-3} \log^4 n)$ work with $O(h + \log n)$ depth for any $1 \leq h \leq n$. The algorithm is correct with high probability and all its comparisons are performed on sums of edge weights on both sides.*

The algorithm proceeds in the following steps:

1. Let $C \subseteq V$ be a set containing each vertex v independently with probability $p = \min(3ch^{-1} \ln n, 1)$ for a sufficiently large constant c.
2. If $|C| > 9cnh^{-1} \ln n$, then terminate.
3. For every vertex $x \in C \cup \{s\}$ and every vertex $v \in V$, compute the shortest h-hop path from x to v in G and its weight $d^h_G(x, v)$.
4. Construct the graph $H = (C \cup \{s\}, (C \cup \{s\})^2, w_H)$ whose set of vertices is $C \cup \{s\}$, whose set of edges is $(C \cup \{s\})^2$ and for every pair of vertices $x, y \in C \cup \{s\}$ the weight of the edge (x, y) is $w_H(x, y) = d^h_G(x, y)$.
5. For every vertex $x \in C$, compute the shortest path from s to x in H and its weight $d_H(s, x)$.
6. For every vertex $t \in V$, set $\delta(t) = \min_{x \in C \cup \{s\}}(d_H(s, x) + d^h_G(x, t))$.

3.1.1 Correctness

In order to prove the correctness of the algorithm, we first observe that as a direct consequence of Lemma 10 the randomly selected vertices in C with high probability hit all lexicographically smallest shortest $\lfloor h/2 \rfloor$-hop paths of the graph.

▶ **Observation 14.** *Consider the collection of sets*

$$\mathcal{S} = \{V(\pi^{\lfloor h/2 \rfloor}(u, v)) \mid u, v \in V \text{ with } d^{\lfloor h/2 \rfloor}_G(u, v) < \infty \text{ and } |E(\pi^{\lfloor h/2 \rfloor}(u, v))| = \lfloor h/2 \rfloor\}$$

containing the vertices of the lexicographically smallest shortest $\lfloor h/2 \rfloor$-hop paths with exactly $\lfloor h/2 \rfloor$ edges between all pairs of vertices. Then, with high probability, C is a hitting set of $\mathcal{S}$ of size at most $9cnh^{-1} \ln n$.

▶ **Lemma 15.** *If G contains no negative cycle, then $\delta(t) = d_G(s, t)$ for every vertex $t \in V$ with high probability.*

Proof. First note that the algorithm incorrectly terminates in Step 2 only with small probability. We now need to show that, for every vertex $t \in V$, $\delta(t) := \min_{x \in C \cup \{s\}}(d_H(s, x) + d^h_G(x, t)) = d_G(s, t)$. First observe that every edge in H corresponds to a path in G (of the same weight). Thus, the value $\delta(t)$ corresponds to some path in G from s to t (of the same weight) which implies that $d_G(s, t) \leq \delta(t)$ (as no path can have weight less than the distance).

Now let $\pi(s, t)$ be the lexicographically smallest shortest path from s to t in G. Subdivide π into consecutive subpaths $\pi_1, \ldots, \pi_k$ such that π_i for $1 \leq i \leq k-1$ has exactly $\lfloor h/2 \rfloor$ edges, and π_k has at most $\lfloor h/2 \rfloor$ edges. Note that if π itself has at most $\lfloor h/2 \rfloor$ edges, then $k = 1$. Since every subpath of a lexicographically smallest shortest path is also a lexicographically smallest shortest path, the paths $\pi_1, \ldots, \pi_k$ are lexicographically smallest shortest paths as well. As the subpaths $\pi_1, \ldots, \pi_{k-1}$ consist of exactly $\lfloor h/2 \rfloor$ edges, each of them is contained in the collection of sets $\mathcal{S}$ of Observation 14. Therefore, each subpath π_i, for $1 \leq i \leq k-1$, contains a vertex $x_i \in C$ with high probability.

Set $x_0 = s$ and $x_k = t$, and observe that for every $0 \leq i \leq k-1$, the subpath of $\pi(s,t)$ from x_i to x_{i+1} is a shortest path from x_i to x_{i+1} with at most h edges and thus $d^h_G(x_i, x_{i+1}) = d_G(x_i, x_{i+1})$. We now get the following chain of inequalities:

$$\begin{aligned} d_G(s,t) = \sum_{0\leq i\leq k-1} d_G(x_i,x_{i+1}) &= \sum_{0\leq i\leq k-1} d^h_G(x_i,x_{i+1}) \\ &= \Big(\sum_{0\leq i\leq k-2} w_H(x_i,x_{i+1})\Big) + d^h_G(x_{k-1},t) \\ &\geq d_H(x_0,x_{k-1}) + d^h_G(x_{k-1},t) \\ &= d_H(s,x_{k-1}) + d^h_G(x_{k-1},t) \\ &\geq \min_{x\in C\cup\{s\}} (d_H(s,x) + d^h_G(x,t)) = \delta(t)\,. \end{aligned}$$

◀

Note that we have formally argued only that the algorithm correctly computes the *distances* from s. It can easily be checked that the shortest paths can be obtained by replacing the edges of H with their corresponding paths in G.

3.1.2 Running Time

▶ **Lemma 16.** *The algorithm above can be implemented with $O(mn\log n + n^3h^{-3}\log^4 n)$ and $O(h+\log n)$ depth such that all its comparisons are performed on sums of edge weights on both sides.*

Proof. Clearly, in Steps 1–2, the algorithm spends $O(m+n)$ work with $O(1)$ depth. Step 3 can be carried out by running h iterations of Bellman-Ford for every vertex $x \in C$ in parallel (see Lemma 8), thus spending $O(|C|\cdot mh)$ work with $O(h)$ depth. Step 4 can be carried out by spending $O(|C|^2)$ work with $O(1)$ depth. Step 5 can be carried out by running the min-plus matrix multiplication based APSP algorithm (see Lemma 6), thus spending $O(|C|^3\log n)$ work with $O(\log n)$ depth. The naive implementation of Step 6 spends $O(n|C|)$ work with $O(|C|)$ depth. Using a bottom-up 'tournament' approach where in each round we pair up all values and let the maximum value of each pair proceed to the next round, this can be improved to work $O(n|C|)$ and depth $O(\log n)$.

It follows that by carrying out the steps of the algorithm sequentially as explained above, the overall work is $O(|C|\cdot mh + |C|^3\log n)$ and the depth is $O(h+\log n)$. As the algorithm ensures that $|C| \leq 9cnh^{-1}\ln n$ for some constant c, the work is $O(mn\log n + n^3h^{-3}\log^4 n)$ and the depth is $O(h+\log n)$. ◀

3.1.3 Extension to Negative Cycle Detection

To check whether a weighted graph $G = (V,E,w)$ contains a negative cycle, we first construct the graph G' (with an additional super-source s') as defined in Lemma 5. We then run the SSSP algorithm of Theorem 13 from s' in G' and set $p(v) = d_{G'}(s',t)$ for every vertex $t \in V$. We then check whether the function p defined in this way is a valid potential function for G testing for every edge $e = (u,v)$ (in parallel) whether $p(u) + w(u,v) \geq p(v)$. If this is the case, then we output that G contains no negative cycle, otherwise we output that G contains a negative cycle.

▶ **Corollary 17.** *There is a randomized algorithm that checks whether a given weighted directed graph contains a negative cycle with $O(mn\log n + n^3h^{-3}\log^4 n)$ work and $O(h+\log n)$ depth for any $1 \leq h \leq n$. The algorithm is correct with high probability and all its comparisons are performed on sums of edge weights on both sides.*

Proof. Constructing the graph G' and checking whether p is a valid potential can both be carried out with $O(m+n)$ work and $O(1)$ depth. Thus, the overall work and depth bounds are asymptotically equal to the SSSP algorithm of Theorem 13.

If G contains no negative cycle, then the SSSP algorithm correctly computes the distances from s' in G'. Thus, the potential p is valid by Lemma 5 and our algorithm correctly outputs that there is no negative cycle. If G contains a negative cycle, then it does not have any valid potential by Lemma 4. Thus, the potential p defined by the algorithm cannot be valid and the algorithm outputs correctly that G contains a negative cycle. ◀

3.2 Finding the Minimum Ratio Cycle

Using the negative cycle detection algorithm as a subroutine, we obtain an algorithm for computing a minimum ratio cycle in time $\tilde{O}(n^{3/2}m^{3/4})$.

▶ **Theorem 18.** *There is a randomized one-sided-error Monte Carlo algorithm for computing a minimum ratio cycle with running time* $O(n^{3/2}m^{3/4}\log^2 n)$.

Proof. By Lemma 2 we can compute the value of the minimum ratio cycle by finding the largest value of λ such that G_λ contains no negative-weight cycle. We want to apply Theorem 1 to find this maximum λ^* by parametric search. As the sequential negative cycle detection algorithm A_s we use Orlin's minimum weight cycle algorithm [46] with running time $T(n,m) = O(mn)$. The parallel negative cycle detection algorithm A_p of Corollary 17 has work $W(n,m) = O(mn\log n + n^3h^{-3}\log^4 n)$ and depth $D(n,m) = O(h+\log n)$, for any choice of $1 \le h \le n$. Any comparison the latter algorithm performs is comparing sums of edge weights of the graph. Since in G_λ edge weights are linear functions in λ, the control flow only depends on testing the sign of degree-1 polynomials in λ. Thus, Theorem 1 is applicable[7] and we arrive at a sequential algorithm for finding the value of the minimum ratio cycle with running time $O(mn\log n(h+\log n) + n^3h^{-3}\log^4 n)$. Finally, to output the minimum ratio cycle and not just its value, we run Orlin's algorithm for finding the minimum weight cycle in G_{λ^*}, which takes time $O(mn)$. By setting $h = n^{1/2}m^{-1/4}\log n$ the overall running time becomes $O(n^{3/2}m^{3/4}\log^2 n)$. ◀

4 Deterministic Algorithm for General Graphs

We now present a deterministic variant of our minimum ratio cycle algorithm, with the same running time as the randomized algorithm up to logarithmic factors.

4.1 Deterministic SSSP and Negative Cycle Detection

We can derandomize our SSSP algorithm by combining a preprocessing step with the parallel hitting set approximation algorithm of [4]. Formally, we will prove the following statement.

▶ **Theorem 19.** *There is a deterministic algorithm that, given a weighted directed graph containing no negative cycles, computes the shortest paths from a designated source vertex s to all other vertices spending* $O(mn\log^2 n + n^3h^{-3}\log^7 n + n^2h\log^{11} n)$ *work with* $O(h+\log^{11} n)$ *depth for any* $1 \le h \le n$.

[7] Formally, Theorem 1 only applies to deterministic algorithms. However, only step 1 of our parallel algorithm is randomized, but this step does not depend on λ. All remaining steps are deterministic. We can thus first perform steps 1 and 2, and invoke Theorem 1 only on the remaining algorithm. The output guarantee then holds with high probability.

From this, using Lemmas 4 and 5 analogously to the proof of Corollary 17, we get the following corollary for negative cycle detection.

▶ **Corollary 20.** *There is a deterministic algorithm that checks whether a given weighted directed graph contains a negative cycle with* $O(mn\log^2 n + n^3h^{-3}\log^7 n + n^2h\log^{11} n)$ *work and* $O(h+\log^{11} n)$ *depth for any* $1 \leq h \leq n$.

Our deterministic SSSP algorithm does the following:

1. For all pairs of vertices $u, v \in V$, compute the shortest $\lfloor h/2 \rfloor$-hop path $\pi^{\lfloor h/2 \rfloor}(u,v)$ from u to v in G.[8]
2. Compute an $O(\log n)$-approximate set cover C of the system of sets $\mathcal{S} = \{V(\pi^{\lfloor h/2 \rfloor}(u,v)) \mid u, v \in V \text{ with } d_G^{\lfloor h/2 \rfloor}(u,v) < \infty \text{ and } |E(\pi^{\lfloor h/2 \rfloor}(u,v))| = \lfloor h/2 \rfloor\}$.
3. Proceed with steps 3 to 6 of the algorithm in Section 3.1.

4.1.1 Correctness

Correctness is immediate: In the previous proof of Lemma 15 we relied on the fact that C is a hitting set of $\mathcal{S}$. In the above algorithm, this property is guaranteed directly.

4.1.2 Running Time

Step 1 can be carried out by running h iterations of the Bellman-Ford algorithm for every vertex $v \in V$. By Lemma 8 this uses $O(mnh)$ work and $O(h)$ depth. We carry out Step 2 by running the algorithm of Theorem 12 to compute an $O(\log n)$-approximate hitting set of $\mathcal{S}$ with work $O(n^2h\log^{11} n)$ and depth $O(\log^{11} n)$. Lemma 10 gives a randomized process that computes a hitting set of $\mathcal{S}$ of expected size $O(nh^{-1}\log n)$. By the probabilistic method, this implies that there exists a hitting set of size $O(nh^{-1}\log n)$. We can therefore use the algorithm of Theorem 12 to compute a hitting set $\mathcal{S}$ of size $O(nh^{-1}\log^2 n)$. The work is $O(n^2h\log^{11} n)$ and the depth is $O(\log^{11} n)$. Carrying out the remaining steps with a hitting set C of size $O(nh^{-1}\log^2 n)$ uses work $O(mh|C| + |C|^3\log n) = O(mn\log^2 n + n^3h^{-3}\log^7 n)$ and depth $O(h+\log n)$. Thus, our overall SSSP algorithm has work $O(mn\log^2 n + n^3h^{-3}\log^7 n + n^2h\log^{11} n)$ and depth $O(h+\log^{11} n)$.

4.2 Minimum Ratio Cycle

We again obtain a minimum ratio cycle algorithm by applying parametric search (Theorem 1). We obtain the same running time bound as for the randomized algorithm.

▶ **Theorem 21.** *There is a deterministic algorithm for computing a minimum ratio cycle with running time* $O(n^{3/2}m^{3/4}\log^2 n)$.

Proof sketch. The proof is analogous to the proof of Theorem 18, with the only exception that we use the deterministic parallel negative cycle detection algorithm of Corollary 20. However, we do not necessarily need to run the algorithm of Theorem 12 to compute an approximate hitting set. Instead we can also run the greedy set cover heuristic (Lemma 11) for this purpose. The reason is that at this stage, the greedy heuristic does not need to perform any comparisons involving the edge weights of the input graph, which are the only operations that are costly in the parametric search technique. This means that finding an approximate hitting

[8] Note that in case there are multiple shortest $\lfloor h/2 \rfloor$-hop paths from u to v, any tie-breaking is fine for the algorithm and its analysis.

set C of size $O(nh^{-1}\log n)$ can be implemented with $O(\sum_{S\in\mathcal{S}}|S|) = O(n^2h)$ work and $O(1)$ comparison depth. Thus, we use a parallel negative cycle detection algorithm A_{p} which has work $W(n,m) = O(mh|C| + |C|^3\log n + n^2h) = O(mn\log n + n^3h^{-3}\log^4 n + n^2h)$ and depth $D(n,m) = O(h+\log n)$, for any choice of $1 \le h \le n$. We thus obtain a sequential minimum ratio cycle algorithm with running time $O(mn\log n + n^3h^{-3}\log^4 n + n^2h + mn\log n(h+\log n))$, for any choice of $1 \le h \le n$. Note that the summands $mn\log n$ and n^2h are both dominated by the last summand $mn\log n(h+\log n)$. Setting $h = n^{1/2}m^{-1/4}\log n$ to optimize the remaining summands, the running time becomes $O(n^{3/2}m^{3/4}\log^2 n)$. ◀

5 Near-Linear Time Algorithm for Constant Treewidth Graphs

In the following we demonstrate how to obtain a nearly-linear time algorithm (in the strongly polynomial sense) for graphs of constant treewidth. We can use the following results of Chaudhuri and Zaroliagis [17] who studied the shortest paths problem in graphs of constant treewidth.[9]

▶ **Theorem 22** ([17]). *There is a deterministic algorithm that, given a weighted directed graph containing no negative cycles, computes a data structure that after $O(n)$ preprocessing time can answer, for any pair of vertices, distance queries in time $O(\alpha(n))$, where $\alpha(\cdot)$ is the inverse Ackermann function. It can also report a corresponding shortest path in time $O(\ell\alpha(n))$, where ℓ is the number of edges of the reported path.*

▶ **Theorem 23** ([16]). *There is a deterministic negative cycle algorithm for weighted directed graphs of constant treewidth with $O(n)$ work and $O(\log^2 n)$ depth.*

We now apply the reduction of Theorem 1 to the algorithm of Theorem 23 to find λ^*, the value of the minimum ratio cycle, in time $O(n\log^3 n)$ (using $T_{\mathrm{s}}(n) = W_{\mathrm{p}}(n) = O(n)$, and $D_{\mathrm{p}}(n) = O(\log^2 n)$). We then use the algorithm of Theorem 22 to find a minimum weight cycle in G_{λ^*} in time $O(n\alpha(n))$: Each edge $e = (u,v)$ together with the shortest path from v to u (if it exists) defines a cycle and we need to find the one of minimum weight by asking the corresponding distance queries. For the edge $e = (u,v)$ defining the minimum weight cycle we query for the corresponding shortest path from v to u. This takes time $O(n)$ as a graph of constant treewidth has $O(n)$ edges. We thus arrive at the following guarantees of the overall algorithm.

▶ **Corollary 24.** *There is a deterministic algorithm that computes the minimum ratio cycle in a directed graph of constant treewidth in time $O(n\log^3 n)$.*

6 Conclusion

We have presented a faster strongly polynomial algorithm for finding a cycle of minimum cost-to-time ratio, a problem which has a long history in combinatorial optimization and recently became relevant in the context of quantitative verification. Our approach combines parametric search with new parallelizable single-source shortest path algorithms and also yields small improvements for graphs of constant treewidth and in the dense regime. The main open problem is to push the running time down to $\tilde{O}(mn)$, nearly matching the strongly polynomial upper bound for the less general problem of finding a minimum mean cycle.

[9] The first result of Chaudhuri and Zaroliagis [17] has recently been complemented with a space-time trade-off by Chatterjee, Ibsen-Jensen, and Pavlogiannis [15] at the cost of polynomial preprocessing time that is too large for our purposes.

References

1 Pankaj K. Agarwal, Micha Sharir, and Sivan Toledo. Applications of parametric searching in geometric optimization. *Journal of Algorithms*, 17(3):292–318, 1994. Announced at SODA'92. doi:10.1006/jagm.1994.1038.

2 Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21(1):136–153, 1980. Announced at ICALP'77. doi:10.1016/0022-0000(80)90046-X.

3 Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

4 Bonnie Berger, John Rompel, and Peter W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. *Journal of Computer and System Sciences*, 49(3):454–477, 1994. Announced at FOCS'89. doi:10.1016/S0022-0000(05)80068-6.

5 Guy E. Blelloch, Yan Gu, Yihan Sun, and Kanat Tangwongsan. Parallel shortest paths using radius stepping. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 443–454, 2016. doi:10.1145/2935764.2935765.

6 Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, Georg Hofferek, Barbara Jobstmann, Bettina Könighofer, and Robert Könighofer. Synthesizing robust systems. *Acta Informatica*, 51(3-4):193–220, 2014. Announced at FMCAD'09. doi:10.1007/s00236-013-0191-5.

7 Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In *International Conference on Computer-Aided Verification (CAV)*, pages 140–156, 2009. doi:10.1007/978-3-642-02658-4_14.

8 Gerth Stølting Brodal, Jesper Larsson Träff, and Christos D. Zaroliagis. A parallel priority queue with constant time operations. *Journal of Parallel and Distributed Computing*, 49(1):4–21, 1998. Announced at IPPS'97. doi:10.1006/jpdc.1998.1425.

9 Steven M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits*. PhD thesis, California Institute of Technology, 1991. Published as technical report CS-TR-91-01.

10 Pavol Cerný, Krishnendu Chatterjee, Thomas A. Henzinger, Arjun Radhakrishna, and Rohit Singh. Quantitative synthesis for concurrent programs. In *International Conference on Computer-Aided Verification (CAV)*, pages 243–259, 2011. doi:10.1007/978-3-642-22110-1_20.

11 Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In *International Conference on Embedded Software (EMSOFT)*, pages 117–133, 2003. doi:10.1007/978-3-540-45212-6_9.

12 Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Symposium on Discrete Algorithms (SODA)*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87.

13 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4):23:1–23:38, 2010. Announced at CSL'08. doi:10.1145/1805950.1805953.

14 Krishnendu Chatterjee, Rasmus Ibsen-Jensen, and Andreas Pavlogiannis. Faster algorithms for quantitative verification in constant treewidth graphs. In *International Conference on Computer-Aided Verification (CAV)*, pages 140–157, 2015. doi:10.1007/978-3-319-21690-4_9.

15 Krishnendu Chatterjee, Rasmus Ibsen-Jensen, and Andreas Pavlogiannis. Optimal reachability and a space-time tradeoff for distance queries in constant-treewidth graphs. In *European Symposium on Algorithms (ESA)*, pages 28:1–28:17, 2016. doi:10.4230/LIPIcs.ESA.2016.28.

16 Shiva Chaudhuri and Christos D. Zaroliagis. Shortest paths in digraphs of small treewdith. Part II: optimal parallel algorithms. *Theoretical Computer Science*, 203(2):205–223, 1998. Announced at ESA'95. doi:10.1016/S0304-3975(98)00021-8.

17 Shiva Chaudhuri and Christos D. Zaroliagis. Shortest paths in digraphs of small treewidth. Part I: sequential algorithms. *Algorithmica*, 27(3):212–226, 2000. Announced at ICALP'95. doi:10.1007/s004530010016.

18 Edith Cohen. Using selective path-doubling for parallel shortest-path computations. *Journal of Algorithms*, 22(1):30–56, 1997. Announced at ISTCS'93. doi:10.1006/jagm.1996.0813.

19 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM*, 47(1):132–166, 2000. Announced at STOC'94. doi:10.1145/331605.331610.

20 Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(m^{10/7} \log W)$ time. In *Symposium on Discrete Algorithms (SODA)*, pages 752–771, 2017. doi:10.1137/1.9781611974782.48.

21 Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34(1):200–208, 1987. Announced at FOCS'84. doi:10.1145/7531.7537.

22 G.B. Dantzig, W. Blattner, and M.R. Rao. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. In P. Rosenstiehl, editor, *Theory of Graphs*, pages 77–84. Dunod, Paris and Gordon and Breach, New York, 1967.

23 Ali Dasdan, Sandy Irani, and Rajesh K. Gupta. Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems. In *Design Automation Conference (DAC)*, pages 37–42, 1999. doi:10.1145/309847.309862.

24 Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. Springer, 2009. doi:10.1007/978-3-642-01492-5.

25 Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972. doi:10.1145/321694.321699.

26 L. R. Ford. Network flow theory. Technical Report P-923, The RAND Corporation, 1956.

27 Bennett Fox. Finding minimal cost-time ratio circuits. *Operations Research*, 17(3):546–551, 1969.

28 Stephan Friedrichs and Christoph Lenzen. Parallel metric tree embedding based on an algebraic view on moore-bellman-ford. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 455–466, 2016. doi:10.1145/2935764.2935777.

29 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

30 Sabih H. Gerez, Sonia M. Heemstra de Groot, and Otto E. Herrmann. A polynomial time algorithm for the computation of the iteration-period bound in recursive data flow graphs. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(1):49–52, 1992. doi:10.1109/81.109243.

31 Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, 1995. Announced at SODA'93. doi:10.1137/S0097539792231179.

32 Manfred v. Golitschek. Optimal cycles in doubly weighted graphs and approximation of bivariate functions by univariate ones. *Numerische Mathematik*, 39(1):65–84, 1982.

33 Mark Hartmann and James B. Orlin. Finding minimum cost to time ratio cycles with small integral transit times. *Networks*, 23(6):567–574, 1993. doi:10.1002/net.3230230607.

34 Alexander T. Ishii, Charles E. Leiserson, and Marios C. Papaefthymiou. An algorithm for the tramp steamer problem based on mean-weight cycles. Technical Report MIT/LCS/TM-457, Massachusetts Institute of Technology, 1991.

35 Kazuhito Ito and Keshab K. Parhi. Determining the minimum iteration period of an algorithm. *VLSI Signal Processing*, 11(3):229–244, 1995. doi:10.1007/BF02107055.

36 David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974. Announced at STOC'73. doi:10.1016/S0022-0000(74)80044-9.

37 Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977. doi:10.1145/321992.321993.

38 Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978. doi:10.1016/0012-365X(78)90011-0.

39 Philip N. Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms*, 25(2):205–220, 1997. Announced at STOC'92. doi:10.1006/jagm.1997.0888.

40 Eugene L. Lawler. Optimal cycles in doubly weighted linear graphs. In P. Rosenstiehl, editor, *Theory of Graphs*, pages 209–214. Dunod, Paris and Gordon and Breach, New York, 1967.

41 Eugene L. Lawler. *Combinatorial Optimization: Network and Matroids*. Holt, Rinehart and Winston, New York, 1976.

42 Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30(4):852–865, 1983. Announced at FOCS'81. doi:10.1145/2157.322410.

43 Ulrich Meyer and Peter Sanders. Δ-stepping: a parallelizable shortest path algorithm. *Journal of Algorithms*, 49(1):114–152, 2003. Announced at ESA'98. doi:10.1016/S0196-6774(03)00076-2.

44 Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 192–201, 2015. doi:10.1145/2755573.2755574.

45 E. F. Moore. The shortest path through a maze. In *International Symposium on the Theory of Switching*, pages 285–292, 1959. doi:10.1137/1.9781611974331.ch87.

46 James B. Orlin. An $O(nm)$ time algorithm for finding the min length directed cycle in a weighted graph. In *Symposium on Discrete Algorithms (SODA)*, pages 1866–1879, 2017. doi:10.1137/1.9781611974782.122.

47 Christos H. Papadimitriou. Efficient search for rationals. *Information Processing Letters*, 8(1):1–4, 1979. doi:10.1016/0020-0190(79)90079-6.

48 Piotr Sankowski. Shortest paths in matrix multiplication time. In *European Symposium on Algorithms (ESA)*, pages 770–778, 2005. doi:10.1007/11561071_68.

49 Hanmao Shi and Thomas H. Spencer. Time-work tradeoffs of the single-source shortest paths problem. *Journal of Algorithms*, 30(1):19–32, 1999. doi:10.1006/jagm.1998.0968.

50 Thomas H. Spencer. Time-work tradeoffs for parallel algorithms. *Journal of the ACM*, 44(5):742–778, 1997. Announced at SODA'91 and SPAA'91. doi:10.1145/265910.265923.

51 Mikkel Thorup. Worst-case update times for fully-dynamic all-pairs shortest paths. In *Symposium on Theory of Computing (STOC)*, pages 112–119, 2005. doi:10.1145/1060590.1060607.

52 Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM Journal on Computing*, 20(1):100–125, 1991. Announced at SPAA'90. doi:10.1137/0220006.

53 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing (STOC)*, pages 664–673, 2014. doi:10.1145/2591796.2591811.

54 Raphael Yuster and Uri Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *Symposium on Foundations of Computer Science (FOCS)*, pages 389–396, 2005. doi:10.1109/SFCS.2005.20.

Simple Greedy Algorithms for Fundamental Multidimensional Graph Problems*

Vittorio Bilò[1], Ioannis Caragiannis[2], Angelo Fanelli[3], Michele Flammini[4], and Gianpiero Monaco[5]

1 **Department of Mathematics and Physics "Ennio De Giorgi", University of Salento, Salento, Italy**
vittorio.bilo@unisalento.it
2 **CTI "Diophantus" & Department of Computer Engineering and Informatics, University of Patras, Patras, Greece**
caragian@ceid.upatras.gr
3 **CNRS (UMR-6211), Caen, France**
angelo.fanelli@unicaen.fr
4 **Gran Sasso Science Institute & DISIM, University of L'Aquila, L'Aquila, Italy**
michele.flammini@univaq.it
5 **DISIM, University of L'Aquila, L'Aquila, Italy**
gianpiero.monaco@univaq.it

Abstract

We revisit fundamental problems in undirected and directed graphs, such as the problems of computing spanning trees, shortest paths, steiner trees, and spanning arborescences of minimum cost. We assume that there are d different cost functions associated with the edges of the input graph and seek for solutions to the resulting multidimensional graph problems so that the p-norm of the different costs of the solution is minimized. We present combinatorial algorithms that achieve very good approximations for this objective. The main advantage of our algorithms is their simplicity: they are as simple as classical combinatorial graph algorithms of Dijkstra and Kruskal, or the greedy algorithm for matroids.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Computations on Discrete Structures, G.2.2 [Graph Theory] Graph Algorithms

Keywords and phrases multidimensional graph problems, matroids, shortest paths, Steiner trees, arborescences

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.125

1 Introduction

We study generalizations of some very well-known combinatorial optimization problems, such as the problem of computing a minimum spanning tree in a graph. In its classical version, we are given an undirected graph with edge costs and the objective is to compute a spanning tree of minimum cost on the graph. We revisit fundamental problems of this kind by assuming that there are d different cost functions associated with the edges of the input graph. Then, a spanning tree has d different cost values, one for each cost function. Our objective is to compute a spanning tree that minimizes a specific aggregate value of these costs.

* This work was partially supported by the project ANR-14-CE24-0007-01 *"CoCoRICo-CoDec"*.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 125; pp. 125:1–125:13

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

At first glance, this is the type of multi-objective (or multi-criteria) optimization problems that arise in many diverse disciplines, including engineering, economics and business, healthcare, and more. Here, our motivation stems from the recently emerging trend of *participatory budgeting* [1]. According to it, optimization problems related to the use of budget for building public facilities are solved taking into account the view the citizens have for the input. In the example above, each of the d different cost functions can be thought of as provided by a single individual. Consistent to this, we will assume that the parameter d is large.

We use the general term *minimum multidimensional resource selection* (MMRS) to refer to the class of problems that we study. In addition to parameter d, an instance of such a problem consists of a set R of resources, a d-dimensional cost vector $\boldsymbol{c}_r$ for each resource $r \in R$, the set of feasible solutions $\mathcal{F}$ (subsets of resources) and an additional parameter $p \geq 1$. The objective of MMRS is to select a feasible solution S so that the quantity $\| \sum_{r \in S} \boldsymbol{c}_r \|_p$ is minimized. Note that the sum inside the norm is d-dimensional and its entries represent the cost of S with respect to the d cost functions. Then, the p-norm is used for aggregating these entries into a single value.

Even though the problem has not been considered before in the general version we just defined it, efficient solutions to some of its variants follow by recent advances on randomized rounding of fractional solutions for linear program relaxations. In contrast to such sophisticated techniques, we insist on deterministic algorithms that are extremely simple. More concretely, we consider the following problems:

- We warm up with MMRS in *matroids*, in which the feasible solutions that form set $\mathcal{F}$ are the bases of a matroid defined over the resources. A typical subproblem is when the resources are the edges of a graph and the feasible solutions correspond to spanning trees of the graph. For MMRS on matroids, we present a variation of the greedy algorithm on matroids (e.g., see [18]) and show that it yields $\mathcal{O}(\min\{p, \log d\})$-approximate solutions.
- *Shortest multidimensional path* (SMP). Again, the resources correspond to edges of a graph and the feasible solutions are subsets of edges that connect two designated nodes. An approximation guarantee of $\mathcal{O}(\min\{p, \log d\})$ is obtained by a Dijkstra-like algorithm.
- *Minimum multidimensional steiner tree* (MMST). Unlike the spanning tree version mentioned above, in MMST the feasible solutions are not matroid bases. Furthermore, the classical trick in the single dimensional case (see, e.g., [19]) of approximating the minimum steiner tree by a minimum spanning tree does not carry over when we have different costs (the cost functions do not necessarily form a metric). Still, we have a Kruskal-like algorithm that uses our shortest multidimensional path algorithm as subroutine and achieves (asymptotically) the same approximation guarantee.
- *Minimum multidimensional arborescence* (MMA). Here, the resources are the edges of a directed graph and the feasible solutions are spanning trees, directed away from a designated root node. We present another simple algorithm that uses our shortest multidimensional path algorithm as a subroutine and prove it to be $\mathcal{O}(\min\{p, \log d\} \cdot \log n)$-approximate, where n is the graph size.

We complement these results with an inapproximability statement. For $p = \infty$, none of the above problems admit a polynomial-time constant approximation algorithm, under standard complexity assumptions. Here we exploit a gap-preserving reduction from the vector scheduling problem which has been proved to be inpproximable in [7].

Our analysis is inspired by the literature on online scheduling and in particular from [2, 5] where the objective is to minimize the p-norm of machine loads. En route, we exploit a nice structural property that is satisfied by feasible solutions of the problems that we study, and implies that greedy solutions for them are efficient.

Related work. The field of multi-objective optimization has traditionally considered similar problems to ours, with approximation algorithms playing a major role. It seems though that the questions considered there are mostly related to approximating the Pareto-curve (i.e., the set of solutions which are not dominated by any other). For instance, Diakonikolas et al. [11] (see also the references therein) study the problem of computing a minimum set of solutions that approximates within a specified accuracy, $\epsilon > 0$, the Pareto curve of bi-objective optimization problems, containing many important widely studied problems such as shortest paths, spanning tree, matching, etc. We notice that these questions are fundamentally different than the ones we study here. Also, they turn out to be computationally meaningful only on instances in which d is a small constant (since otherwise the problem becomes notoriously hard due to fact that the Pareto-curve becomes huge [16]). Instead, we are interested in many different cost functions.

Chekuri et al. [8] (see also [9]) study the problem of solving (or, better, approximating the optimal solution of) *minimax integer programs* subject to a matroid constraint; this is essentially what we call MMRS on matroids with a value of infinity for parameter p. Chekuri et al. [8] present a $\mathcal{O}(\log d/\log\log d)$-approximation algorithm for this problem that exploits sophisticated randomized rounding techniques. We remark that our bound is slightly higher than theirs but the advantage of our result is in the simplicity of the algorithm. A special case is covered in [4], where the computation of a spanning tree minimizing the maximum number of times its edges cross a given set of cuts is considered.

Other investigations related to our setting are the multi-budgeted optimization problems. There are d different cost functions defined over the set of resources. In addition, there are d budget values that constrain each of the d costs of a solution. The objective is to compute a feasible solution whose budget violation factor is as small as possible across all cost dimensions. These problems have been tackled using sophisticated approaches such as Lagrangian relaxations combined with various technical properties of the underlying combinatorial structure, and linear programming together with iterative rounding techniques. Such approaches were used to develop PTASes for spanning trees [14, 17], shortest paths [13, 15], and matchings and matroid intersection [3] with $d = 2$. Grandoni et al. [12] consider d-budgeted versions of classical problems. They show PTASes for spanning trees, matroid bases, and bipartite matchings. Moreover they get a deterministic approximation scheme for d-budgeted matchings in general graphs. We emphasize that the authors of [12] use linear programming formulations and iterative rounding techniques that work for constant values of d only. Finally, Chekuri et al. [10] give a randomized PTAS for matroid intersection and matchings with any fixed number of budget constraints.

Roadmap. We begin with some necessary mathematical background in Section 2. Our main lemma is presented in Section 3. Then, Sections 4-7 are devoted to the each of the four problems mentioned above. We conclude with our inapproximability result in Section 8.

2 Mathematical Background

We summarize definitions and simple properties of p-norms and matroids. For an integer $d \geq 1$, define $[d] = \{1, 2, \ldots, d\}$ and $\mathbf{0}^d$ as the vector $(0, \ldots, 0)^T \in \mathbb{R}^d$. We denote by $\overline{\mathbb{R}} = \mathbb{R}_{\geq 1} \cup \{\infty\}$ the set of reals that are higher than 1, extended with the value ∞.

We later exploit the following two properties of the function $f(x) = x^t$ for every $t \geq 1$.

▶ **Lemma 1.** *For every $x, y, h \geq 0$ with $y \geq x$ and $t \geq 1$, $(x + h)^t - x^t \leq (y + h)^t - y^t$.*

▶ **Lemma 2.** *Given $x \geq 0$, $t \geq 1$, and n non-negative real values $h_1, \ldots, h_n$,*

$$\sum_{i\in[n]} \left((x+h_i)^t - x^t\right) \leq \left(x + \sum_{i\in[n]} h_i\right)^t - x^t.$$

The first property comes by observing that $(x+h)^t - x^t$ has non-negative derivative with respect to x when $t \geq 1$, while the second one is due to convexity of the monomial x^t (see [6] for a proof).

For a vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $p \in \overline{\mathbb{R}}$, the value $\|\boldsymbol{x}\|_p = \left(\sum_{i\in[d]} x_i^p\right)^{1/p}$ is called the *p-norm* of $\boldsymbol{x}$. We recall two fundamental properties possessed by p-norms.

▶ Property 3. For every $\boldsymbol{x} \in \mathbb{R}^d$ and $p, p' \in \overline{\mathbb{R}}$ such that $p' \leq p$, $\|\boldsymbol{x}\|_p \leq \|\boldsymbol{x}\|_{p'}$.

▶ Property 4 (Minkowsi's Inequality). For every $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ and $p \in \overline{\mathbb{R}}$, $\|\boldsymbol{x}+\boldsymbol{y}\|_p \leq \|\boldsymbol{x}\|_p + \|\boldsymbol{y}\|_p$.

The next lemma shows how to use the logarithmic norm to approximate all other p-norms; its proof follows easily by the definitions.

▶ **Lemma 5.** *For every $\boldsymbol{x} \in \mathbb{R}^d$ and $p \in \overline{\mathbb{R}}$, $\|\boldsymbol{x}\|_{\ln d} \leq e\|\boldsymbol{x}\|_p$.*

A *matroid* is a pair $M = (R, \mathcal{X})$ such that R is a finite set, called the *ground set*, and $\mathcal{X}$ is a family of subsets of R with the following properties:

1. $\emptyset \in \mathcal{X}$,
2. if $X \in \mathcal{X}$ and $Y \subset X$, then $Y \in \mathcal{X}$ (*hereditary property*),
3. if $X, Y \in \mathcal{X}$ and $|X| > |Y|$, then there exists $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{X}$ (*independent set exchange property*).

A *basis* for matroid M is a set $B \in \mathcal{X}$ such that $B \cup \{x\} \notin \mathcal{X}$ for every $x \in R \setminus B$. The independent set exchange property implies that all bases of M have the same cardinality which is called the *rank* of M and is denoted by $r(M)$.

For every two bases $B_1, B_2 \in \mathcal{X}$, denote by $G(B_1 \Delta B_2)$ the bipartite graph (V, E) such that $V = (B_1 \setminus B_2) \cup (B_2 \setminus B_1)$ and $E = \{\{e_1, e_2\} : e_1 \in B_1 \setminus B_2, e_2 \in B_2 \setminus B_1, B_1 \setminus \{e_1\} \cup \{e_2\} \in \mathcal{X}\}$. We shall make extensive use of the following fundamental result (see [18]).

▶ Proposition 6. There exists a perfect matching in the graph $G(B_1 \Delta B_2)$.

Given an undirected graph $G = (V, E)$, let $\mathcal{X}$ be the family of all subsets of E which do not contain cycles. The pair $M = (E, \mathcal{X})$ is a matroid and is called the *graphic matroid* defined over G. The set of bases for M is the set of all spanning trees for G, so that $r(M) = |V| - 1$.

3 Problem Statement and the PAID Property

The minimum multidimensional resource selection (MMRS) problem is a collection of instances of the form $I = (R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F}, p)$, where R is a set of resources such that each resource $r \in R$ has an associated d-dimensional cost vector $\boldsymbol{c}_r \in \mathbb{R}^d_+$, $\mathcal{F} \subseteq 2^R \setminus \emptyset$ is a set of feasible solutions, and $p \in \overline{\mathbb{R}}$. For a subset of resources $S \subseteq R$, define its multidimensional load as $\boldsymbol{\ell}(S) = \sum_{r\in S} \boldsymbol{c}_r$ and denote by $\ell_i(S)$ its ith element. An optimal solution for I is any solution belonging to $\operatorname{argmin}_{S\in\mathcal{F}} \left\{\|\boldsymbol{\ell}(S)\|_p\right\}$, that is, any feasible solution minimizing the p-norm of its multidimensional load. Denote by $\mathsf{OPT}(I) = \|\boldsymbol{\ell}(S^*)\|_p$, where $S^* \in \operatorname{argmin}_{S\in\mathcal{F}} \left\{\|\boldsymbol{\ell}(S)\|_p\right\}$, the p-norm of the multidimensional load of an optimal solution for I.

We shall denote by $\mathsf{MMRS}(p)$ the natural restriction of the MMRS obtained by fixing the value of p. When referring to an instance $I \in \mathsf{MMRS}(p)$, we remove the value of p from the description of I and simply write $I = (R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F})$.

Given a set X and an integer $k \geq 1$, a *partial k-decomposition* of X is an ordered family of k sets $(X_1, \ldots, X_k)$ such that $\bigcup_{i\in[k]} X_i \subseteq X$. A *$\beta$-intersecting partial k-decomposition* of X is a partial k-decomposition of X $(X_1, \ldots, X_k)$ such that $|\{i \in [k] : x \in X_i\}| \leq \beta$ for every $x \in X$, that is, no element of X occurs in more than β components of the decomposition. A k-decomposition of X is a partial *k-decomposition* of X $(X_1, \ldots, X_k)$ such that $\bigcup_{i\in[k]} X_i = X$. A *$k$-partition* of X is a 1-intersecting k-decomposition of X.

▶ **Definition 7.** Fix an instance $I \in \mathsf{MMRS}(p)$. A feasible solution S for I has the *pairwise β-intersecting decomposition* property (henceforth, β-PAID property) if there exist a k-decomposition of S $(S_1, \ldots, S_k)$ and a β-intersecting partial k-decomposition of an optimal solution S^* for I $(S_1^*, \ldots, S_k^*)$ such that, for every $i \in [k]$,

$$\|\boldsymbol{\ell}(S_{\leq i})\|_p \leq \|\boldsymbol{\ell}(S_{\leq i-1}) + \boldsymbol{\ell}(S_i^*)\|_p\,, \tag{1}$$

where $S_{\leq i} = \bigcup_{j\in[i]} S_j$ and $S_{\leq 0} = \emptyset$.

The importance of the PAID property is captured by the following result.

▶ **Lemma 8.** *Fix an instance $I \in \mathsf{MMRS}(p)$. If a feasible solution S for I possesses the β-PAID property, then $\|\boldsymbol{\ell}(S)\|_p \leq \frac{\beta p}{\ln 2}\mathsf{OPT}(I)$.*

Proof. We get

$$\begin{aligned}
\left(\|\boldsymbol{\ell}(S)\|_p\right)^p &= \sum_{j\in[d]} \ell_j(S)^p = \sum_{i\in[k]} \left(\sum_{j\in[d]} \ell_j(S_{\leq i})^p - \sum_{j\in[d]} \ell_j(S_{\leq i-1})^p\right)\\
&\leq \sum_{i\in[k]} \left(\sum_{j\in[d]} \left(\ell_j(S_{\leq i-1}) + \ell_j(S_i^*)\right)^p - \sum_{j\in[d]} \ell_j(S_{\leq i-1})^p\right)\\
&= \sum_{j\in[d]}\sum_{i\in[k]} \left(\left(\ell_j(S_{\leq i-1}) + \ell_j(S_i^*)\right)^p - \ell_j(S_{\leq i-1})^p\right)\\
&\leq \sum_{j\in[d]}\sum_{i\in[k]} \left(\left(\ell_j(S) + \ell_j(S_i^*)\right)^p - \ell_j(S)^p\right)\\
&\leq \sum_{j\in[d]} \left(\left(\ell_j(S) + \sum_{i\in[k]} \ell_j(S_i^*)\right)^p - \ell_j(S)^p\right)\\
&\leq \sum_{j\in[d]} \left(\left(\ell_j(S) + \beta\ell_j(S^*)\right)^p - \ell_j(S)^p\right)\\
&= \sum_{j\in[d]} \left(\ell_j(S) + \beta\ell_j(S^*)\right)^p - \sum_{j\in[d]} \ell_j(S)^p\\
&\leq \left(\|\boldsymbol{\ell}(S)\|_p + \beta\|\boldsymbol{\ell}(S^*)\|_p\right)^p - \left(\|\boldsymbol{\ell}(S)\|_p\right)^p.
\end{aligned}$$

The first inequality follows by raising both sides of inequality (1) to p. The second and third inequalities follow from Lemmas 1 and 2, respectively. The fourth inequality holds since $(S_1^*, \ldots, S_k^*)$ is a β-intersecting partial k-decomposition of S^*. The fifth inequality follows by Minkowski's inequality (by raising both of its sides to p).

By rearranging, we obtain $\left(2^{1/p} - 1\right)\|\boldsymbol{\ell}(S)\|_p \leq \beta\|\boldsymbol{\ell}(S^*)\|_p$, which implies

$$\|\boldsymbol{\ell}(S)\|_p \leq \frac{\beta\|\boldsymbol{\ell}(S^*)\|_p}{2^{1/p} - 1} = \frac{\beta\|\boldsymbol{\ell}(S^*)\|_p}{e^{\ln 2/p} - 1} \leq \frac{\beta p}{\ln 2}\|\boldsymbol{\ell}(S^*)\|_p,$$

Algorithm 1 $(M, d, (\boldsymbol{c}_r)_{r\in R}, p)$

$S_0 \leftarrow \emptyset$
$i \leftarrow 0$
while $S_{\leq i}$ is not a basis for M **do**
  $i \leftarrow i + 1$
  $C_i \leftarrow \{x \in R : S_{\leq i-1} \cup \{x\} \in \mathcal{X}\}$
  $S_i \leftarrow \operatorname{argmin}_{x\in C_i} \|\boldsymbol{\ell}(S_{\leq i-1} \cup \{x\})\|_p$
end while
$S \leftarrow S_{\leq i}$
return S

where the last inequality comes from the fact that $e^x \geq x + 1$ for every $x \geq 0$. Since S^* is an optimal solution for I, the claim follows. ◀

By exploiting Lemma 5, we get the following approximability result; the proof is omitted due to lack of space.

▶ **Lemma 9.** *Let S be an α-approximate solution to an instance $I = (R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F}) \in$ MMRS$(\ln d)$. Then, S is an $O(\alpha)$-approximate solution to the instance $\overline{I} = (R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F}) \in$ MMRS(p) with $p \geq \ln d$.*

By putting all together, we get the following general approximation theorem.

▶ **Theorem 10.** *Let A be an algorithm which, for every instance $I \in$ MMRS, computes a feasible solution for I possessing the β-PAID property. Then, A approximates* MMRS *within a factor of $O(\beta \cdot \min\{p, \log d\})$.*

Proof. Fix an instance $I = (R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F}, p) \in$ MMRS. We can use algorithm A to obtain a feasible solution S for I possessing the β-PAID property. By Lemma 8, S is an $O(p\beta)$-approximate solution for I. Moreover, we can use algorithm A to obtain a feasible solution S for the instance $I' = (R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F}, \ln d)$ possessing the β-PAID property. By Lemma 8, S is an $O(\beta \log d)$-approximate solution for I' so that, by Lemma 9, S is also an $O(\beta \log d)$-approximate solution for I. ◀

4 MMRS on Matroids

As a warmup application of our technique, we first consider instances $(R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F}, p) \in$ MMRS such that $\mathcal{F}$ is the set of bases of a matroid $M = (R, \mathcal{X})$. We propose a simple greedy algorithm (Algorithm 1) to approximate MMRS in this case.

The following lemma characterizes the approximation guarantee achieved by Algorithm 1.

▶ **Lemma 11.** *Fix an instance $I = (R, d, (\boldsymbol{c}_r)_{r\in R}, \mathcal{F}, p) \in$ MMRS such that $\mathcal{F}$ is the set of bases of a matroid $M = (R, \mathcal{X})$. Algorithm 1 returns a feasible solution for I possessing the* 1-PAID *property.*

Proof. The fact that Algorithm 1 terminates by returning a feasible solution $S \in \mathcal{F}$ (i.e., a basis for M) follows from the classical analysis of the greedy algorithm for matroids. Set $k = r(M)$ and let S^* be an optimal solution to I. Define $R(S, S^*) = S \cap S^*$ and consider graph $G(S \Delta S^*)$. By Proposition 6, there exists a bijective function $f : S \setminus R(S, S^*) \to S^* \setminus R(S, S^*)$.

Note that Algorithm 1 implicitly defines a k-partition $(S_1, \ldots, S_k)$ of S. Now define the k-partition $(S_1^*, \ldots, S_k^*)$ of S^* such that, for every $i \in [k]$,

$$S_i^* = \begin{cases} S_i & \text{if } S_i \in R(S, S^*), \\ f(S_i) & \text{if } S_i \notin R(S, S^*). \end{cases}$$

Fix an index $i \in [k]$. We claim that, at the ith iteration of the while-loop of Algorithm 1, $S_{\leq i-1} \cup \{S_i^*\} \in C_i$. In fact, if this is not the case, it must be $S_{\leq i-1} \cup \{S_i^*\} \notin \mathcal{X}$. However, by the definition of f, we have that $S \setminus \{S_i\} \cup \{S_i^*\} \in \mathcal{X}$ which, by the hereditary property of matroids, implies that $S_{\leq i-1} \cup \{S_i^*\} \in \mathcal{X}$: a contradiction. Now, given that $S_i^* \in C_i$, the greedy choice performed at line 6 of Algorithm 1 implies that $\|\boldsymbol{\ell}(S_{\leq i})\|_p \leq \|\boldsymbol{\ell}(S_{\leq i-1} \cup S_i^*)\|_p \leq \|\boldsymbol{\ell}(S_{\leq i-1}) + \boldsymbol{\ell}(S_i^*)\|_p$. ◀

By combining the above lemma with Theorem 10, we obtain the following result.

▶ **Theorem 12.** *Algorithm 1 approximates* MMRS *on matroids within a factor of* $O(\min\{p, \log d\})$.

5 Shortest Multidimensional Path

Given a directed graph $G = (V, E)$, in which every edge $e \in E$ is associated with a d-dimensional weight $\boldsymbol{c}_e \in \mathbb{R}_+^d$, a pair $(s, t) \in V^2$ of source-destination nodes, and value $p \in \overline{\mathbb{R}}$, the shortest multidimensional path (SMP) problem is the restriction of MMRS to instances with $R = E$ and $\mathcal{F} = \{S \subseteq E : S \text{ is an } (s,t)\text{-path in } G\}$.

For our purposes, we shall need to solve instances of the SMP when dealing with other MMRSs on graphs. For such a reason, we shall define an approximation algorithm for the SMP which requires more general input parameters than the ones needed to solve the SMP.

Towards this end, consider the multidimensional generalization of Dijkstra's algorithm, denoted as Algorithm 2, defined in the following. It takes as input the graph G, the integer d, the pair of nodes (s, t), the value p, and a set of edges E' which may contain either edges in E and edges not in E, and makes use of the data structures PATH and DISTANCE. Given a node $v \in V$, PATH is an array such that PATH$[v]$ contains a path connecting s to v, and DISTANCE is an array such that DISTANCE$[v]$ contains the value $\|\boldsymbol{\ell}(\text{PATH}[v] \cup E')\|_p$.

The following lemma characterizes the approximation guarantee achieved by Algorithm 2.

▶ **Lemma 13.** *Fix an instance $I = (G, d, s, t, p) \in$ SMP. Then, Algorithm 2, executed with parameters G, d, s, t, p and $\emptyset$, returns a feasible solution for I possessing the* 1-PAID *property.*

Proof. The fact that, when executed with parameters G, d, s, t, p and $\emptyset$, Algorithm 2 terminates by returning a set of edges $S = \text{PATH}[t]$ inducing an (s,t)-path in G follows from the classical analysis of Dijkstra's algorithm. Hence, we only need to show that S possesses the 1-PAID property.

Let S^* be an optimal solution to I. For a node v and a path P, let $pred_P(v)$ be the predecessor of node v along P. A node v is a *merging node* for S and S^* if (i) $v \in \{s, t\}$ or (ii) v occurs along both S and S^* and $pred_S(v) \neq pred_{S^*}(v)$. Denote by $M(S, S^*) = (s = v_0, v_1, \ldots, v_j = t)$ the sequence of merging nodes for S and S^* numbered according to the order in which they occur along S. A node $v_i \in M(S, S^*)$ is *redundant* if v_i occurs along S^* before some other merging node v_j with $j < i$. Denote by $\overline{M}(S, S^*) = (s = \overline{v}_0, \overline{v}_1, \ldots, \overline{v}_k = t)$ the sequence of nodes obtained from $M(S, S^*)$ by removing all the redundant ones (see Figure 1 for an illustrating example).

Algorithm 2 (G, d, s, t, p, E')

```
1: for each v ∈ V do
2:     PATH[v] ← ∅
3:     DISTANCE[v] ← +∞
4: end for
5: DISTANCE[s] ← ‖ℓ(E′)‖_p
6: Q ← V
7: while Q is not empty do
8:     v ← argmin_{u∈Q}{DISTANCE[u]}
9:     for each (v, u) ∈ E do
10:        if DISTANCE[u] > ‖ℓ(PATH[v] ∪ (v, u) ∪ E′)‖_p then
11:            PATH[u] ← PATH[v] ∪ (v, u)
12:            DISTANCE[u] ← ‖ℓ(PATH[v] ∪ (v, u) ∪ E′)‖_p
13:        end if
14:    end for
15:    Q ← Q \ {v}
16: end while
17: return PATH[t]
```

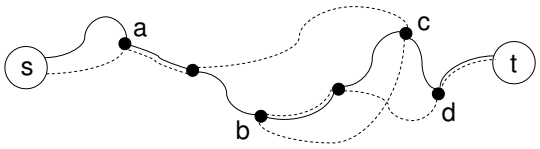


Figure 1 The definition of non-redundant merging nodes used in the proof of Lemma 13. The solid lines represent the (s,t)-path S returned by Algorithm 2, while the dashed ones represent the optimal solution S^* (solid and dashed lines are drawn adjacently when some set of edges are shared by S and S^*). We have $M(S,S^*) = \{s,a,b,c,d,t\}$ and $\overline{M}(S,S^*) = \{s,a,b,d,t\}$ since node c is redundant.

Let $(S_1,\ldots,S_k)$ and $(S_1^*,\ldots,S_k^*)$ be the k-partitions of S and $\overline{S}^*$, respectively, such that, for every $i \in [k]$, S_i (resp., S_i^*) is the set of edges connecting $\overline{v}_{i-1}$ to $\overline{v}_i$ in S (resp., S^*).

The well-known semantics of Dijkstra's algorithm guarantees that, for every $i \in [k]$, the set of edges S_i satisfies the inequality

$$\mathsf{DISTANCE}[\overline{v}_i] \leq \|\boldsymbol{\ell}(\mathsf{PATH}[\overline{v}_{i-1}] \cup S_i^*)\|_p,$$

where, by construction, $\mathsf{DISTANCE}[\overline{v}_i] = \|\boldsymbol{\ell}(S_{\leq i})\|_p$ and

$$\|\boldsymbol{\ell}(\mathsf{PATH}[\overline{v}_{i-1}] \cup S_i^*)\|_p = \|\boldsymbol{\ell}(S_{\leq i-1} \cup S_i^*)\|_p \leq \|\boldsymbol{\ell}(S_{\leq i-1}) + \boldsymbol{\ell}(S_i^*)\|_p .$$

Hence, the claim follows. ◀

By combining Lemma 13 with Theorem 10, we obtain the following result.

▶ **Theorem 14.** *Algorithm 2 approximates* SMP *within a factor of* $O(\min\{p, \log d\})$.

We conclude this section by showing a fundamental lemma that will allow us to use Algorithm 2 as a subroutine of approximation algorithms for other MMRSs defined on graphs.

▶ **Lemma 15.** *Given an instance $I \in$ MMRS, let $(S_1, \ldots, S_k)$ be a k-decomposition of a feasible solution S for I and $(S_1^*, \ldots, S_k^*)$ be a β-intersecting partial k-decomposition of an optimal solution S^* for I. If, for every $i \in [k]$, there exists an instance $I_i = (G_i, d, s_i, t_i, p) \in$ SMP such that S_i is an (s_i, t_i)-path in G_i computed by using Algorithm 2 executed with parameters G_i, d, s_i, t_i, p and $S_{\leq i-1}$, and S_i^* is an (s_i, t_i)-path in G_i, then S possesses the β-PAID property.*

Proof. The proof is a direct extension of the proof of Lemma 13. In fact, it suffices to exploit the decomposition technique used therein within the decompositions $(S_1, \ldots, S_k)$ and $(S_1^*, \ldots, S_k^*)$. Toward this end, denoting by $(S_{i,1}, \ldots, S_{i,h_i})$ and $(S_{i,1}^*, \ldots, S_{i,h_i}^*)$ the h_i-partitions of S_i and S_i^*, respectively, that are obtained as in the proof of Lemma 13 and setting $m = \sum_{\in[k]} h_i$, we have that

$$(S_{1,1}, \ldots, S_{1,h_1}, \ldots, S_{k,1}, \ldots, S_{k,h_k}) = (T_1, \ldots, T_m)$$

is an m-decomposition of S and that

$$(S_{1,1}^*, \ldots, S_{1,h_1}^*, \ldots, S_{k,1}^*, \ldots, S_{k,h_k}^*) = (T_1^*, \ldots, T_m^*)$$

is a β-intersecting partial m-decomposition of S^*. By the same arguments used in the proof of Lemma 13, we obtain that $\|\boldsymbol{\ell}(T_{\leq i})\|_p \leq \|\boldsymbol{\ell}(T_{\leq i-1} \cup T_i^*)\|_p \leq \|\boldsymbol{\ell}(T_{\leq i-1}) + \boldsymbol{\ell}(T_i^*)\|_p$ for each $i \in [m]$, thus proving the claim. ◀

6 Minimum Multidimensional Steiner Tree

Given an undirected graph $G = (V, E)$, in which every edge $e \in E$ is associated with a d-dimensional weight $\boldsymbol{c}_e \in \mathbb{R}_+^d$, a set of $r+1$ required nodes $N = \{v_1, \ldots, v_{r+1}\} \subseteq V$, and a value $p \in \overline{\mathbb{R}}$, the minimum multidimensional Steiner tree (MMST) problem is the restriction of MMRS to instances with $R = E$ and such that $\mathcal{F}$ is the set of all trees in G whose set of nodes contains N.

We propose Algorithm 3, a Kruskal-like algorithm which takes as input the graph G, the integer d, the set of required nodes N and the value p, and uses Algorithm 2 and the functions *set*, *merge* and *prune* as subroutines. Given a required node v and an h-partition $P = \{P_1, P_2, \ldots, P_h\}$ of N, function *set* returns the set $P_i \in P$ such that $v \in P_i$; given a partition $P = \{P_1, P_2, \ldots, P_h\}$ of N and two sets $P_i, P_j \in P$, function *merge* returns the partition of N obtained from P by merging P_i and P_j; finally, given a set of edges S, function *prune* returns a maximal set of edges $S' \subseteq S$ not inducing cycles in G.

The following lemma characterizes the approximation guarantee achieved by Algorithm 3.

▶ **Lemma 16.** *Fix an instance $I = (G, d, N, p) \in$ MMST. Then, Algorithm 3 returns a feasible solution for I possessing the* 2-PAID *property.*

Proof. By the well-known semantics of Kruskal's algorithm, we have that the while-loop at lines 4-12 of Algorithm 3 is executed exactly r times so that $S = (S_1, \ldots, S_r)$ induces a subgraph of G spanning N and $\overline{S} = prune(S)$ is a Steiner tree spanning N. This implies that the set of edges $\overline{S}$ returned by Algorithm 3 is a feasible solution for I. We shall prove that S possesses the 2-PAID property which, given that $\overline{S} \subset S$, implies the claim.

Let S^* be an optimal solution to I. Our proof is based on the following idea: for every $i \in [r]$, we associate a path $g(i) \in S^*$ to path $\pi(s_i, t_i) = S_i$ such that every edge in S^* appears at most twice along all sets of paths $\{g(1), \ldots, g(r)\}$. We shall prove that, using this function, we obtain an r-decomposition of S and a 2-intersecting r-decomposition of

Algorithm 3 (G, d, N, p)

1: $S_0 \leftarrow \emptyset$
2: $i \leftarrow 1$
3: $P_i \leftarrow \{\{v_1\}, \ldots, \{v_{r+1}\}\}$
4: **while** $P_i \neq N$ **do**
5: **for** each $(s,t) \in N^2 : set(s, P_i) \neq set(t, P_i)$ **do**
6: $\pi(s,t) \leftarrow$ Algorithm 2$(G, d, s, t, p, S_{\leq i-1})$
7: **end for**
8: $(s_i, t_i) \leftarrow \text{argmin}_{(s,t) \in N^2 : set(s,P_i) \neq set(t,P_i)} \{\|\boldsymbol{\ell}(S_{\leq i-1} \cup \pi(s,t))\|_p\}$
9: $S_i \leftarrow \pi(s_i, t_i)$
10: $P_{i+1} \leftarrow merge(P_i, set(P_i, s_i), set(P_i, t_i))$
11: $i \leftarrow i + 1$
12: **end while**
13: **return** $prune(S_{\leq i-1})$

S^*, meeting the conditions required in order to apply Lemma 15; the claim will then follow directly.

Towards this end, given two required nodes $u, v \in N$, let $\Sigma_{uv} = \{\pi^1_{uv}, \ldots, \pi^{p_{uv}}_{uv}\}$ denote the set of all (u,v)-paths in G. Define $\widetilde{G} = (N, \widetilde{E})$ as the multi-graph such that there are p_{uv} edges $\{\widetilde{e}^1_{uv}, \ldots, \widetilde{e}^{p_{uv}}_{uv}\}$ between every pair of nodes $u, v \in N$, with $\boldsymbol{c}_{\widetilde{e}^i_{uv}} = \boldsymbol{\ell}(\pi^i_{uv})$ for every $i \in [p_{uv}]$; so, there is a cost-preserving bijection between edges in $\widetilde{G}$ and paths in G. Given a path π in G, denote by $\widetilde{e}(\pi)$ its correspondent edge in $\widetilde{G}$ and, vice-versa, given an edge $\widetilde{e}$ of $\widetilde{G}$, denote by $\rho(\widetilde{e})$ its correspondent path in G.

We observe the following facts:

1. S induces a spanning tree $T(S)$ for $\widetilde{G}$ defined as $T(S) = \{\widetilde{e}(\pi(s_i, t_i)) \in \widetilde{E} : i \in [r]\}$.
2. S^* induces a spanning tree $T(S^*)$ for $\widetilde{G}$ defined as follows: let $(v_1, \ldots, v_{r+1})$ be the ordered sequence of the $r+1$ nodes in N listed according to the order in which appear along a Depth First Search of S^* starting from an arbitrary required node $v_1 \in N$, then $T(S^*) = \{\widetilde{e}(\pi^*_i) : i \in [r]\}$, where π^*_i is the (v_i, v_{i+1})-path in S^*. It is well-known that every edge in S^* occurs at most twice in the set of paths $\{\widetilde{e}(\pi^*_1), \ldots, \widetilde{e}(\pi^*_r)\}$.
3. Since both $T(S)$ and $T(S^*)$ are bases of the graphic matroid defined over $\widetilde{G}$, by applying Proposition 6, there exists a bijection $f : T(S) \setminus T(S^*) \to T(S^*) \setminus T(S)$.

Let us define a function $g : [r] \to S^*$ such that, for every $i \in [r]$,

$$g(i) = \begin{cases} \pi(s_i, t_i) & \text{if } \widetilde{e}(\pi(s_i, t_i)) \in T(S^*), \\ \rho(f(\widetilde{e}(\pi(s_i, t_i)))) & \text{if } \widetilde{e}(\pi(s_i, t_i)) \notin T(S^*). \end{cases}$$

For every $i \in [r]$, set $S^*_i = g(i)$. We have that $(S_1, \ldots, S_r)$ is an r-decomposition of S and $(S^*_1, \ldots, S^*_r)$ is a 2-intersecting r-decomposition of S^*. Our aim now is to apply Lemma 15.

Towards this end, fix an index $i \in [r]$ and denote by s^*_i and t^*_i the two endpoints of path $g(i)$. We observe that it must be $set(s^*_i, P_i) \neq set(t^*_i, P_i)$. Indeed, if this is not the case, then $(T(S) \setminus \widetilde{e}(\pi(s_i, t_i)) \cup \widetilde{e}(g(i))$ cannot be a basis of the graphic matroid defined over $\widetilde{G}$.

At the ith iteration of the while-loop at lines 4-12 of Algorithm 3, P_i represents the set of connected components of G induced by the set of edges $S_{\leq i-1}$. Define $\overline{G}_i$ as the multi-graph obtained from G by contracting the connected components containing $set(s_i, P_i)$ and $set(s^*_i, P_i)$ into a super-node $\overline{s}_i$ and the connected components containing $set(t_i, P_i)$ and $set(t^*_i, P_i)$ into a super node $\overline{t}_i$. Let G_i be the graph obtained from $\overline{G}_i$ by splitting every

multi-edge $e = \{u_i, u_j\}$ of weight $\boldsymbol{c}_e$ into two edges $\{u_i, u_{ij}\}$ and $\{u_{ij}, u_j\}$ of weights $\boldsymbol{c}_e$ and $\mathbf{0}^d$, respectively.

If $set(s_i, P_i) = set(s_i^*, P_i)$ and $set(t_i, P_i) = set(t_i^*, P_i)$, we have that $(G_i, \overline{s}_i, \overline{t}_i, p)$ is an instance of the SMP meeting the conditions of Lemma 15, by which, we obtain the claim. If $set(s_i, P_i) = set(s_i^*, P_i)$ and $set(t_i, P_i) = set(t_i^*, P_i)$ does not hold, since G is an undirected graph, we may assume without loss of generality that $set(s_i^*, P_i) \neq set(t_i, P_i)$ and $set(s_i, P_i) \neq set(t_i^*, P_i)$ (in fact, if one of the two inequalities does not hold, we can exchange the role of s_i^* and t_i^* and have both of them satisfied).

Now observe that there is a cost-preserving bijection b between the set of paths connecting any node in $set(s_i, P_i) \cup set(s_i^*, P_i)$ to any node in $set(t_i, P_i) \cup set(t_i^*, P_i)$ in G and the set of $(\overline{s}_i, \overline{t}_i)$-paths in G_i. Moreover, by the assumption $set(s_i, P_i) \neq set(t_i^*, P_i)$, it follows that path $b(S_i)$ is an $(\overline{s}_i, \overline{t}_i)$-path in G_i; similarly, by the assumption $set(s_i^*, P_i) \neq set(t_i, P_i)$, it follows that path $b(g_i)$ is an $(\overline{s}_i, \overline{t}_i)$-path in G_i.

Thus, in order to apply Lemma 15 and obtain the claim, we need to prove that there are suitable tie breaking rules for which $b(S_i)$ is the output of Algorithm 2 when executed with parameters G_i, d, $\overline{s}_i$, $\overline{t}_i$, p and $S_{\leq i-1}$ on the instance $(G_i, d, \overline{s}_i, \overline{t}_i, p) \in$ SMP. Assume, by way of contradiction, that for any possible tie breaking rule, Algorithm 2 never returns an $(\overline{s}_i, \overline{t}_i)$-path π_i such that $\pi_i = b(S_i)$. This implies that there exists an $(\overline{s}_i, \overline{t}_i)$-path π_i^* such that $\|\boldsymbol{\ell}(S_{\leq i-1} \cup \pi_i^*)\|_p < \|\boldsymbol{\ell}(S_{\leq i-1} \cup \pi_i)\|_p$ by which we obtain $\|\boldsymbol{\ell}(S_{\leq i-1} \cup b^{-1}(\pi_i^*))\|_p < \|\boldsymbol{\ell}(S_{\leq i-1} \cup b^{-1}(\pi_i))\|_p$, where $b^{-1}(\pi_i^*)$ is some (s,t)-path in G with $set(s, P_i) \neq set(t, P_i)$. This contradicts $S_i = \text{argmin}_{(s,t) \in N^2 : set(s,P_i) \neq set(t,P_i)} \{\|\boldsymbol{\ell}(S_{\leq i-1} \cup \pi(s,t))\|_p\}$. ◀

By combining Lemma 16 with Theorem 10, we obtain the following result.

▶ **Theorem 17.** *Algorithm 3 approximates* MMST *within a factor of* $O(\min\{p, \log d\})$.

7 Minimum Multidimensional Arborescence

Given a directed connected graph $G = (V, E)$, with $|V| = n$, in which every edge $e \in E$ is associated with a d-dimensional weight $\boldsymbol{c}_e \in \mathbb{R}_+^d$, a node $s \in V$ and a value $p \in \overline{\mathbb{R}}$, the minimum multidimensional arborescence (MMA) problem is the restriction of MMRS to instances with $R = E$ and such that $\mathcal{F}$ is the set of all the directed trees in G rooted at s. Recall that a directed tree $T \subset E$ rooted at s is a set of $n-1$ edges such that, for every $t \in V$, there exists a directed (s,t)-path in T.

A *rooted weakly connected component* of G is a subgraph $H = (V', E')$ of G possessing at least one root, that is, a node from which it is possible to reach any other node in V' by following a direct path in E'. Call the *representative node* of a rooted weakly connected component any of its roots. When the set of roots of a rooted weakly connected component H contains s, the representative node of H is always assumed to be s.

We propose Algorithm 4 which, starting from the set of rooted weakly connected components of G obtained by considering all nodes in V as singletons, repeatedly merges rooted weakly connected components of G until an arborescence rooted at s is obtained. Given a set of edges S, function *repr* first computes the set C of weakly connected components of G induced by S, and then computes a representative for every element of C. Observe that, by our assumption, $s \in repr(C)$ for every set of rooted weakly connected components C. Given a set of edges S and a node u, function *nodes* computes the set of nodes belonging to the weakly connected component containing u. Finally, given a set of edges S, function *prune* returns a maximal subset of S not inducing cycles in G.

The following lemma characterizes the approximation guarantee achieved by Algorithm 4. Its proof is omitted due to lack of space.

Algorithm 4 (G, d, s, p)

$S_0 \leftarrow \emptyset$
$C_1 \leftarrow V$
$i \leftarrow 1$
while $C_i \neq \{s\}$ **do**
  $S_i \leftarrow S_{i-1}$
  for each $u \in C_i \setminus \{s\}$ **do**
    for each $v \in V \setminus nodes(u, S_{i-1})$ **do**
      $\pi(v, u) \leftarrow Algorithm\ 2(G, d, v, u, p, S_i)$
    end for
    $\delta(u) \leftarrow \text{argmin}_{v \in V \setminus nodes(u, S_{i-1})}\{\|\boldsymbol{\ell}(S_i \cup \pi(v, u))\|_p\}$
    $S_i \leftarrow S_i \cup \pi(\delta(u), u)$
  end for
  $C_{i+1} \leftarrow repr(S_i)$
  $i \leftarrow i + 1$
end while
return $prune(S_{i-1})$

▶ **Lemma 18.** *Fix an instance $I = (G, d, s, p) \in$ MMA. Then, Algorithm 4 returns a feasible solution for I possessing the $O(\log n)$-PAID property.*

By combining Lemma 18 with Theorem 10, we obtain the following result.

▶ **Theorem 19.** *Algorithm 4 approximates* MMA *within a factor of $O(\log n \cdot \min\{p, \log d\})$.*

8 An Inapproximability Result

We conclude by complementing our positive algorithmic results with the following hardness statement.

▶ **Theorem 20.** *For every constant $\kappa \geq 1$, the problems* MMRS(∞) *on matroids,* SMP(∞), MMST(∞), *and* MMA(∞) *cannot be approximated up to a factor κ unless* NP=ZPP.

Proof. Our proof defines a simple approximation-preserving reduction from the Vector Scheduling Problem VSP. An instance of the VSP is defined by n tasks to be scheduled on m identical machines. Every task i has a d-dimensional load vector $\boldsymbol{c}_i \in \mathbb{R}^d_+$ and the objective is to minimize the load over all machines and all dimensions. Chekuri and Kanna [7] showed that, for every constant $\kappa \in \mathbb{R}$, this problem cannot be approximated up to a factor κ unless NP=ZPP.

Given an instance of the VSP, we define an undirected multi-graph $G = (V, E)$ with $n+1$ nodes and mn edges defined as follows: $V = \{v_0, v_1, \ldots, v_n\}$, and for every $i \in [n-1]$ there are m edges $e_i^1, \ldots, e_i^m \in E$ connecting nodes v_{i-1} and v_i such that, for every $j \in [m]$, edge e_i^j has a (dm)-dimensional cost

$$\boldsymbol{c}(e_i^j) = (\underbrace{\mathbf{0}^d, \ldots, \mathbf{0}^d}_{j-1 \text{ times}}, \boldsymbol{c}_i, \underbrace{\mathbf{0}^d, \ldots, \mathbf{0}^d}_{m-j \text{ times}}).$$

Observe that every schedule of the n tasks to the m machines corresponds to a (v_0, v_n)-path in G and viceversa. Moreover, the objective value of the two solutions is exactly the same. Since G can be easily translated into a graph G' by splitting every multi-edge into two edges of the same total cost, we have that the hardness result for the VSP extends also

to $\mathsf{SMP}(\infty)$. Given that every spanning tree for G' is a (v_0, v_n)-path in G', the hardness result extends to $\mathsf{MMRS}(\infty)$ on matroids and to $\mathsf{MMST}(\infty)$ when the set of required nodes contains both v_0 and v_n. Finally, by directing every edge in E from v_{i-1} to v_i, we obtain the same hardness result for $\mathsf{MMA}(\infty)$. ◀

References

1 https://www.participatorybudgeting.org/

2 B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter. Load balancing in the L_p norm. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 383–391, 1995.

3 A. Berger, V. Bonifaci, F. Grandoni, G. Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Mathematical Programming*, 128(1-2): 355–372, 2011.

4 V. Bilò, V. Goyal, R. Ravi, and M. Singh. On the crossing spanning tree problem. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Problems (APPROX)*, LNCS 3122, pages 51–60, 2004.

5 I. Caragiannis. Better bounds for online load balancing on unrelated machines. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 972–982, 2008.

6 I. Caragiannis. Efficient coordination mechanisms for unrelated machine scheduling. *Algorithmica*, 66(3): 512–540, 2013.

7 C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM Journal on Computing*, 33(4): 837–851, 2004.

8 C. Chekuri, J. Vondrák and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 575–584, 2010.

9 C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding for matroid polytopes and applications. *arXiv*: 0909.4348, 2009.

10 C. Chekuri, J. Vondrák, and R. Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1080–1097, 2011.

11 I. Diakonikolas, M. Yannakakis. Small approximate Pareto sets for biobjective shortest paths and other problems. *SIAM Journal on Computing*, 39(4): 1340-1371, 2009.

12 F. Grandoni, R. Ravi, M. Singh, and R. Zanklusen. New approaches to multi-objective optimization. *Mathematical Programming*, 146(1): 525–554, 2014.

13 R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operation Research*, 17(1): 36–42, 1992.

14 R. Hassin, Asaf Levin. An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection. *SIAM Journal on Computing*, 33(2): 261–268, 2004.

15 D. Lorenz, D. Raz. A simple efficient approximation scheme for the restricted shortest paths problem. *Operations Research Letters*, 28: 213–219, 2001.

16 C. H. Papadimitriou, M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 86–92, 2000.

17 R. Ravi, M. Goemans. The constrained minimum spanning tree problem. In *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 66–75, 1996.

18 A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency. Volume B, Matroids, Trees, Stable Sets*. Springer, 2003.

19 V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

Stochastic k-Server: How Should Uber Work?*†

Sina Dehghani[1], Soheil Ehsani[2], Mohammad Hajiaghayi[3], Vahid Liaghat[4], and Saeed Seddighin[5]

1 **University of Maryland, College Park, MD, USA**
dehghani@cs.umd.edu
2 **University of Maryland, College Park, MD, USA**
ehsani@cs.umd.edu
3 **University of Maryland, College Park, MD, USA**
hajiagha@cs.umd.edu
4 **Facebook, Menlo Park, CA, USA**
vliaghat@gmail.com
5 **University of Maryland, College Park, MD, USA**
saeedrez@cs.umd.edu

Abstract

In this paper we study a stochastic variant of the celebrated k-server problem. In the k-server problem, we are required to minimize the total movement of k servers that are serving an online sequence of t requests in a metric. In the stochastic setting we are given t independent distributions $\langle P_1, P_2, \ldots, P_t \rangle$ in advance, and at every time step i a request is drawn from P_i.

Designing the optimal online algorithm in such setting is NP-hard, therefore the emphasis of our work is on designing an approximately optimal online algorithm. We first show a structural characterization for a certain class of *non-adaptive* online algorithms. We prove that in general metrics, the best of such algorithms has a cost of no worse than three times that of the optimal online algorithm. Next, we present an integer program that finds the optimal algorithm of this class for any arbitrary metric. Finally by rounding the solution of the linear relaxation of this program, we present an online algorithm for the stochastic k-server problem with an approximation factor of 3 in the line and circle metrics and factor of $O(\log n)$ in a general metric of size n. In this way, we achieve an approximation factor that is independent of k, the number of servers.

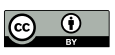
Moreover, we define the *Uber* problem, motivated by extraordinary growth of online network transportation services. In the Uber problem, each demand consists of two points -a source and a destination- in the metric. Serving a demand is to move a server to its source and then to its destination. The objective is again minimizing the total movement of the k given servers. We show that given an α-approximation algorithm for the k-server problem, we can obtain an $(\alpha + 2)$-approximation algorithm for the Uber problem. Motivated by the fact that demands are usually highly correlated with the time (e.g. what day of the week or what time of the day the demand has arrived), we study the *stochastic Uber* problem. Using our results for stochastic k-server we can obtain a 5-approximation algorithm for the stochastic Uber problem in line and circle metrics, and a $O(\log n)$-approximation algorithm for general metrics.

Furthermore, we extend our results to the correlated setting where the probability of a request arriving at a certain point depends not only on the time step but also on the previously arrived requests.

1998 ACM Subject Classification Computer science education

* A full version of the paper is available at https://arxiv.org/abs/1705.05755.

† Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant.



44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 126; pp. 126:1–126:14

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

Keywords and phrases k-server, stochastic, competitive ratio, online algorithm, Uber

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.126

1 Introduction

The k-server problem is one of the most fundamental problems in online computation that has been extensively studied in the past decades. In the k-server problem we have k mobile servers on a metric space $\mathcal{M}$. We receive an online sequence of t requests where the i^{th} request is a point $r_i \in \mathcal{M}$. Upon the arrival of r_i, we need to move a server to r_i, at a cost equal to the distance from the current position of the server to r_i. The goal is to minimize the total cost of serving all requests.

Manasse, McGeoch, and Sleator [31] introduced the k-server problem as a natural generalization of several online problems, and a building block for other problems such as the metrical task systems. They considered the adversarial model, in which the online algorithm has no knowledge of the future requests. Following the proposition of Sleator and Tarjan [34], they evaluate the performance of an online algorithm using competitive analysis. In this model, an online algorithm ALG is compared to an *offline* optimum algorithm OPT which is aware of the entire input in advance. For a sequence of requests ρ, let $|\mathrm{ALG}(\rho)|$ and $|\mathrm{OPT}(\rho)|$ denote the total cost of ALG and OPT for serving ρ. An algorithm is *c-competitive* if for every ρ, $|\mathrm{ALG}(\rho)| \leq c\,|\mathrm{OPT}(\rho)| + c_0$ where c_0 is independent of ρ.

Manasse *et al.* [31] showed a lower bound of k for the competitive ratio of any deterministic algorithm in any metric space with at least $k+1$ points. The celebrated *k-server conjecture* states that this bound is tight for general metrics. For several years the known upper bounds were all exponential in k, until a major breakthrough was achieved by Koutsoupias and Papadimitriou [29], who showed that the so-called *work function algorithm* is $(2k-1)$-competitive. Proving the tight competitive ratio has been the "holy grail" of the field in the past two decades. This challenge has led to the study of the problem in special spaces such as the uniform metric (also known as the paging problem), line, circle, and trees metrics (see [15, 16] and references therein). We also refer the reader to Section 1.3 for a short survey of randomized algorithms, particularly the recent result of Bansal, Buchbinder, Madry, and Naor [7] which achieves the competitive ratio of $O(\log^3 n \log^2 k)$ for discrete metrics that comprise n points.

The line metric (or Euclidean 1-dimensional metric space) is of particular interest for developing new ideas. Chrobak, Karloof, Payne, and Vishwnathan [15] were the first to settle the conjecture in the line by designing an elegant k-competitive algorithm. Chrobak and Larmore [16] generalized this approach to tree metrics. Later, Bartal and Koutsoupias [10] proved that the work function algorithm is also k-competitive in line. Focusing on the special case of $k = 2$ in line, Bartal *et al.* [9] show that, using randomized algorithms, one can break the barrier of lower bound k by giving a 1.98-competitive algorithm for the case where we only have two servers.

Despite the strong lower bounds for the k-server problem, there are heuristics algorithms that are *constant* competitive in practice. For example, for the paging problem- the special case of uniform metric- the least recently used (LRU) strategy is shown to be experimentally constant competitive (see Section 1.3). In this paper we present an algorithm an run it on real world data to measure its empirical performance. In particular we use the distribution of car accidents obtained from road safety data. Our experiments illustrate our algorithm is performing even better in practice. See the full version of the paper for more details.

The idea of comparing the performance of an online algorithm (with zero-knowledge of the future) to the request-aware offline optimum has led to crisp and clean solutions. However, that is not without its downsides. The results in the online model are often very pessimistic leading to theoretical guarantees that are hardly comparable to experimental results. Indeed, one way to tighten this gap is to use stochastic information about the input data as we describe in this paper.

We should also point out that the competitive analysis is not the only possible or necessarily the most suitable approach for this problem. Since the distributions from which the input is generated are known, one can use dynamic programming (or enumeration of future events) to derive the optimal movement of servers. Unfortunately, finding such an optimal online solution using the distributions is an NP-hard problem [1], thus the dynamic programming or any other approach takes exponential time. This raises the question that how well one can perform in comparison to the best online solution. In the rest of the paper we formally define the model and address this question.

A natural and well-motivated generalization of k-server is to assume the demands are two points instead of just one, consisting of a source and a destination. To serve a demand we need to move a server to the source and then move it to the destination. We call this problem the *Uber* problem. One can see, the Uber problem is the same as k-server when the sources and the destinations are the same. We also show that, given an α-approximation algorithm for the k-server problem, we can obtain a $(\alpha + 2)$-approximation algorithm for the Uber problem. Thus our results for k-server also apply to the Uber problem.

1.1 The Stochastic Model

In this paper, we study the *stochastic k-server problem* where the input is not chosen adversarially, but consists of draws from given probability distributions. This problem has lots of applications such as network transportations and equipment replacement in data centers. The current mega data centers contain hundreds of thousands of servers and switches with limited life-span. For example servers usually retire after at most three years. The only efficient way to scale up the maintenance in data centers is by automation, and robots are designed to handle maintenance tasks such as repairs or manual operations on servers. The replacement process can be modeled as requests that should be satisfied by robots, and robots can be modeled as servers. This problem also has applications in physical networks. As an example, suppose we model a shopping service (e.g. Google Express) as a k-server problem in which we receive an online sequence of shopping requests for different stores. We have k shopping cars (i.e., servers) that can serve the requests by traveling to the stores. It is quiet natural to assume that on a certain time of the week/day, the requests arrive from a distribution that can be discovered by analyzing the history. For example, an Uber request is more likely to be from suburb to midtown in the morning, and from midtown to suburb at night. We formalize this stochastic information as follows.

For every $i \in [1 \cdots t]$, a discrete probability distribution P_i is given in advance from which request r_i will be drawn at time step i. The distributions are chosen by the adversary and are assumed to be independent but not necessarily identical. This model is inspired by the

[1] Reduction from k-median to Stochastic k-server: to find the k median of set S of vertices, one can construct an instance of stochastic k-server with $t = 1$ and $P_1(v) = 1/|S|$ for every $v \in S$. The best initialization of the servers gives the optimum solution to k-median of S.

well-studied model of *prophet inequalities* [2] [30, 25]. As mentioned before, the case of line metric has proven to be a very interesting restricted case for studying the k-server problem. In this paper, we focus mainly on the class of line metric though our results carry over to circle metric and general metrics as well.

In the adversarial model, the competitive ratio seems to be the only well-defined notion for analyzing the performance of online algorithms. However, in the presence of stochastic information, one can derive a much better benchmark that allows us to make fine-grained distinctions between the online algorithms. We recall that in the offline setting, for a class of algorithms $\mathcal{C}$, the natural notion to measure the performance of an algorithm $\text{ALG} \in \mathcal{C}$ is the *approximation ratio* defined as the worse case ratio of $|\text{ALG}|$ to $|\text{OPT}(\mathcal{C})|$ where $\text{OPT}(\mathcal{C})$ is the optimal algorithm in the class. In this paper, we also measure the performance of an online algorithm by its approximation ratio– compared to the *optimal online solution*. We note that given distributions $P_1, \ldots, P_t$, one can iteratively compute the optimal online solution by solving the following exponential-size dynamic program: for every $i \in [0 \cdots t]$ and every possible placement A of k servers (called a *configuration*) on the metric, let $\tau(i, A)$ denote the minimum expected cost of an online algorithm for serving the first i requests and then moving the servers to configuration A. Note that $\tau(i, A)$ can inductively be computed via the following recursive formula

$$\tau(i, A) = \min_B \tau(i-1, B) + \mathrm{E}_{r_i \sim P_i} [\text{min. distance from } B \text{ to } A \text{ subject to serving } r_i] \ ,$$

where $\tau(0, A)$ is initially zero for every A.

1.2 Our Results [3]

Our first main result is designing a constant approximation algorithm in the line metric when the distributions for different time steps are not necessarily identical.

▶ **Theorem 1.** *There exists a 3-approximation online algorithm for the stochastic k-server problem in the line metric. The running time is polynomial in k and the sum of the sizes of the supports of input distributions. The same guarantee holds for the circle metric.*

For the general metric, we present an algorithm with a logarithmic approximation guarantee.

▶ **Theorem 2.** *There exists a $O(\log n)$-approximation online algorithm for the stochastic k-server problem in a general metric of size n.*

We prove the theorems using two important structural results. The first key ingredient is a general reduction from class of online algorithms to a restricted class of *non-adaptive algorithms* while losing only a constant factor in the approximation ratio. Recall that a configuration is a placement of k-servers on the metric. We say an algorithm ALG is *non-adaptive* if it follows the following procedure: ALG pre-computes a sequence of configurations $A_0, A_1, \ldots, A_t$. We start by placing the k-servers on A_0. Upon the arrival of r_i, (i) we move the servers to configuration A_i; next (ii) we move the closest server s to r_i; and finally (iii) we return s to its original position in A_i. We first prove the following structural result.

---

[2] In the prophet inequality setting, given (not necessarily identical) distributions $P_1, \ldots, P_t$, an online sequence of values $x_1, \ldots, x_n$ where x_i is drawn from P_i, an onlooker has to choose one item from the succession of the values, where x_i is revealed at step i. The onlooker can choose a value only at the time of arrival. The goal is to maximize the chosen value.

[3] In the interest of space, we have omitted some of the proofs. We refer the reader to the full version of the paper in order to see all of the proofs.

▶ **Theorem 3.** *For the stochastic k-server problem in the general metric, the optimal non-adaptive online algorithm is within 3-approximation of the optimal online algorithm.*

Using the aforementioned reduction, we focus on designing the optimal non-adaptive algorithm. We begin by formulating the problem as an integer program. The second ingredient is to use the relaxation of this program to formalize a natural *fractional variant* of the problem. In this variant, a configuration is a fractional assignment of *server mass* to the points of the metric such that the total mass is k. To serve a request at point r_i, we need to move some of the mass to have at least one amount of server mass on r_i. The cost of moving the server mass is naturally defined as the integral of the movement of infinitesimal pieces of the server mass. By solving the linear relaxation of the integer program, we achieve the optimal fractional non-adaptive algorithm. We finally prove Theorems 1 and 2 by leveraging the following rounding techniques. The rounding method in line has been also observed by Türkoglu [35]. We provide the proof for the case of line in Section 5 for the sake of completeness. The rounding method for general metrics is via the well-known embedding of a metric into a distribution of well-separated trees while losing a logarithmic factor in the distortion. Bansal *et al.* [7] use a natural rounding method similar to that of Blum, Burch, and Kalai [12] to show that any fractional k-server movement on well-separated trees can be rounded to an integral counterpart by losing only a constant factor.

▶ **Theorem 4** (first proven in [35]). *Let* ALG_f *denote a fractional k-server algorithm in the line, or circle. One can use* ALG_f *to derive a randomized integral algorithm* ALG *such that for every request sequence σ,* $\mathrm{E}\left[|\mathrm{ALG}(\sigma)|\right] = |\mathrm{ALG}_f(\sigma)|$. *The expectation is over the internal randomness of* ALG. *Furthermore, in the stochastic model* ALG *can be derandomized.*

▶ **Theorem 5** (proven in [7]). *Let* ALG_f *denote a fractional k-server algorithm in any metric. One can use* ALG_f *to derive a randomized integral algorithm* ALG *such that for every request sequence σ,* $\mathrm{E}\left[|\mathrm{ALG}(\sigma)|\right] \leq O(\log n)\,|\mathrm{ALG}_f(\sigma)|$.

We also show that having an α-approximation algorithm for k-server, we can obtain a $(\alpha+2)$-approximation for the Uber problem, using a simple reduction.

▶ **Theorem 6.** *Let* ALG *denote an α-approximation algorithm for k-server. One can use* ALG *to derive a $(\alpha+2)$-approximation algorithm for the Uber problem.*

Proof. Consider an instance of the Uber problem I_U. Let s_i and t_i denote the i-th source and destination, respectively. We generate an instance of the k-server problem I_k by removing every t_i from I_U. In other words the demands are s_i's. We use ALG to provide a solution for I_U as follows. For satisfying the i-th demand, we use ALG to move a server to s_i. Then using the shortest path from s_i to t_i, we move that server to t_i and then return it back to s_i. Let OPT_U and OPT_k denote the cost of the optimal solutions for I_U and I_k, respectively. Let $d(s_i, t_i)$ denote the distance of t_i from s_i in the metric. Let C denote the total movement of the servers. We have,

$$\mathrm{OPT}_U \geq \mathrm{OPT}_k\,.$$

$$\mathrm{OPT}_U \geq \sum_i d(s_i, t_i).$$

$$C \leq \alpha\,\mathrm{OPT}_k + 2\sum_i d(s_i, t_i) \leq (\alpha+2)\,\mathrm{OPT}_U\,.$$ ◀

1.3 Further Related Work

The randomized algorithms often perform much better in the online paradigm. For the k-server problem, a lower bound of $\Omega(\log k)$ is shown by [28] for the competitive ratio of randomized algorithms in most common metrics. Despite the exponential gap, compared to the lower bound of deterministic algorithms, very little is known about the competitiveness of randomized algorithms. In fact, the only known algorithms with competitive ratios below k, work either in the uniform metric (also known as the paging problem [21, 32, 2, 8]), a metric comprising $k+1$ points [23], and two servers on the line [9]. Two decades after the introduction of the k-server problem, a major breakthrough was achieved by Bansal *et al.* [7] in discrete metrics with sub-exponential size. If $\mathcal{M}$ comprise n points, their randomized algorithm achieves a competitive ratio of $O(\log^3 n \log^2 k)$.

The case of uniform metric has been extensively studied under various stochastic models motivated by the applications in computer caching. Koutsoupias and Papadimitriou [29] consider two refinements of the competitive analysis for server problems. First, they consider the *diffuse adversary* model. In this model, at every step i the adversary chooses a distribution D_i over the uniform metric of the paging problem. Then the i^{th} request is drawn from D_i which needs to be served. The distribution D_i is not known to the online algorithm and it may depend on the previous requests. However, in their paper, they consider the case wherein it is guaranteed that for every point p, $D_i(p) \leq \epsilon$ for a small enough ϵ; i.e., the next request is not predictable with absolute certainty for the adversary. The results of Koutsoupias and Papadimitriou and later Young [36] shows that the optimum competitive ratio in this setting is close to $1 + \Theta(k\epsilon)$.

The second refinement introduced in [29] restricts the optimal solution to having lookahead at most ℓ. Hence, one can define a *comparative ratio* which indicates the worst-case ratio of the cost of the best online solution to the best solution with lookahead ℓ. They show that for the k-server problem, and more generally the metrical task system problem, there are online algorithms that admit a comparative ratio of $2\ell + 1$; for some instances this ratio is tight.

Various other models of restricting the adversary (access graph model [14, 26, 22], fault rate model [27, 6, 19], etc) have also been considered for the paging problem (see [33, 11] and references therein for a further survey of these results). Unfortunately, many of the stochastic settings considered for the paging problem do not seem to have a natural generalization beyond the uniform metric setting. For example, in the diffuse adversary model, most of the studied distributions do not weaken the adversary in the general metric. In this paper, we look for polynomial-time *approximation* algorithms in the class of online algorithms that have access to the distributions.

We would like to mention that various online problems have been previously considered under prophet inequality model or i.i.d. model (where all distributions are identical). The maximum matching problem, scheduling, and online network design has been extensively studied in these models(see e.g. [4, 3, 5, 17, 1, 18]). In the graph connectivity problems, Garg, Gupta, Leonardi,and Sankowski [24] consider the online variants of Steiner tree and several related problems under the i.i.d. stochastic model. In the adversarial model, there exists an $\Omega(\log n)$ lower bound on the competitive ratio of any online algorithm, where n is the number of demands. However, Garg *et al.* show that under the i.i.d. assumption, these problems admit online algorithms with constant or $O(\log \log n)$ competitive ratios. We refer the reader to the excellent book by Borodin and El-Yaniv [13] for further study of online problems.

2 Preliminaries

In this section we formally define the stochastic k-server problem. The classical k-server problem is defined on a metric $\mathcal{M}$ which consists of points that could be infinitely many. For every two points x and y in metric $\mathcal{M}$, let $d(x, y)$ denote the distance of x from y which is a symmetric function and satisfies the triangle inequality. More precisely for every three points x, y, and z we have

$$d(x, x) = 0 \tag{1}$$
$$d(x, y) = d(y, x) \tag{2}$$
$$d(x, y) + d(y, z) \geq d(x, z). \tag{3}$$

In the k-server problem the goal is to place k servers on k points of the metric, and move these servers to satisfy the requests. We refer to every placement of the servers on the metric points by a *configuration*. Let $\rho = \langle r_1, r_2, \ldots, r_t \rangle$ be a sequence of requests, the goal of the k-server problem is to find configurations $\langle A_0, A_1, A_2, \ldots, A_t \rangle$ such that for every i there exists a server on point r_i in configuration A_i. We say such a list of configurations is *valid* for the given list of requests. A valid sequence of configurations is optimal if $\sum d(A_{i-1}, A_i)$ is minimized where $d(X, Y)$ stands for the minimum cost of moving servers from configuration X to configuration Y. An optimal sequence $\langle A_0, A_1, \ldots, A_t \rangle$ of configurations is called an optimal offline solution of $\mathrm{OFKS}(\mathcal{M}, \rho)$ when ρ is known in advance. We refer to the optimal cost of such movements with $|\mathrm{OFKS}(\mathcal{M}, \rho)| = \sum d(A_{i-1}, A_i)$.

We also define the notion of *fractional configuration* as an assignment of the metric points to non-negative real numbers. More precisely, each number specifies a mass of fractional server on a point. Every fractional solution adheres to the following condition: The total sum of the values assigned to all points is exactly equal to k. Analogously, a fractional configuration serves a request r_i if there is a mass of size at least 1 of server assigned to point r_i. An offline fractional solution of the k-server problem for a given sequence of requests ρ is defined as a sequence of fractional configurations $\langle A_0, A_1, \ldots, A_t \rangle$ such that A_i serves r_i.

In the online k-server problem, however, we are not given the whole sequence of requests in the beginning, but we will be informed of every request once its realization is drawn. An algorithm $\mathcal{A}$ is an online algorithm for the k-server problem if it reports a configuration A_0 as an initial configuration and upon realization of every request r_i it returns a configuration A_i such that $\langle A_0, A_1, \ldots, A_i \rangle$ is valid for $\langle r_1, r_2, \ldots, r_i \rangle$. If $\mathcal{A}$ is deterministic, it generates a unique sequence of configurations for every sequence of requests. Let $\mathcal{A}(\mathcal{M}, \rho)$ be the sequence that $\mathcal{A}$ generates for requests in ρ and $|\mathcal{A}(\mathcal{M}, \rho)|$ denote its cost.

In the online stochastic k-server problem, in addition to metric $\mathcal{M}$, we are also given t independent probability distributions $\langle P_1, P_2, \ldots, P_t \rangle$ which show the probability that every request r_i is realized on a point of the metric at each time. An algorithm $\mathcal{A}$ is an online algorithm for such a setting, if it generates a configuration for every request r_i not solely based on $\langle r_1, r_2, \ldots, r_i \rangle$ and $\langle A_0, A_1, \ldots, A_{i-1} \rangle$ but also with respect to the probability distributions. Similarly, we define the cost of an online algorithm $\mathcal{A}$ for a given sequence of requests ρ with $|\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t \rangle)|$. We define the expected cost of an algorithm $\mathcal{A}$ on metric $\mathcal{M}$ and with probability distributions $\langle P_1, P_2, \ldots, P_t \rangle$ by

$$|\mathcal{A}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t \rangle)| = \mathbb{E}_{\forall i, r_i \sim P_i} |\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t \rangle)|.$$

For every metric $\mathcal{M}$ and probability distributions $\langle P_1, P_2, \ldots, P_t \rangle$ we refer to the online algorithm with the minimum expected cost by $\mathrm{OPT}_{\mathcal{M}, \langle P_1, P_2, \ldots, P_t \rangle}$.

An alternative way to represent a solution of the k-server problem is as a vector of configurations $\langle B_0, B_1, \ldots, B_t\rangle$ such that B_i does not necessarily serve request r_i. The cost of such solution is equal to $\sum d(B_{i-1}, B_i) + \sum 2d(B_i, r_i)$ where $d(B_i, r_i)$ is the minimum distance of a server in configuration B_i to request r_i. The additional cost of $2d(B_i, r_i)$ can be thought of as moving a server from B_i to serve r_i and returning it back to its original position. Thus, every such representation of a solution can be transformed to the other representation. Similarly, $d(B_i, r_i)$ for a fractional configuration B_i is the minimum cost which is incurred by placing a mass 1 of server at point r_i. We use letter B for the configurations of such solutions throughout the paper.

In this paper the emphasis is on the stochastic k-server problem on the line metric. We define the line metric $\mathcal{L}$ as a metric of points from $-\infty$ to $+\infty$ such that the distance of two points x and y is always equal to $|x-y|$. Moreover, we show that deterministic algorithms are as powerful as randomized algorithms in this setting, therefore we only focus on deterministic algorithms in this paper. Thus, from here on, we omit the term deterministic and every time we use the word algorithm we mean a deterministic algorithm unless otherwise is explicitly mentioned.

3 Structural Characterization

Recall that an online algorithm $\mathcal{A}$ has to fulfill the task of reporting a configuration A_i upon arrival of request r_i based on $\langle A_0, A_1, \ldots, A_{i-1}\rangle$, $\langle r_1, r_2, \ldots, r_i\rangle$, and $\langle P_1, P_2, \ldots, P_t\rangle$. We say an algorithm $\mathcal{B}$ is *request oblivious*, if it reports configuration B_i regardless of request r_i. As such, $\mathcal{B}$ generates configurations $\langle B_0, B_1, \ldots, B_t\rangle$ for a sequence of requests $\langle r_1, r_2, \ldots, r_t\rangle$ and the cost of such configuration is $\sum d(B_{i-1}, B_i) + \sum 2d(B_i, r_i)$. More precisely, no matter what request r_i is, $\mathcal{B}$ will generate the same configuration for a given list of past configurations $\langle B_0, B_1, \ldots, B_{i-1}\rangle$, a given sequence of past requests $\langle r_1, r_2, \ldots, r_{i-1}\rangle$, and the sequence of probability distributions $\langle P_1, P_2, \ldots, P_t\rangle$. In the following we show that every online algorithm $\mathcal{A}$ can turn into a request oblivious algorithm $\mathcal{B}_\mathcal{A}$ that has a cost of at most $|3\mathcal{A}(\mathcal{M}, \rho, \langle P_1, P_2, \ldots, P_t\rangle)|$ for a given sequence of requests ρ.

▶ **Lemma 7.** *Let $\mathcal{A}$ be an online algorithm for the stochastic k-server problem. For any metric $\mathcal{M}$, there exists a request oblivious algorithm $\mathcal{B}_\mathcal{A}$ such that*

$$|\mathcal{B}_\mathcal{A}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)| \leq 3|\mathcal{A}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)|.$$

An immediate corollary of Lemma 7 is that the optimal request oblivious algorithm has a cost of at most $|3\,\mathrm{OPT}_{\mathcal{M},\langle P_1,P_2,\ldots,P_t\rangle}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t\rangle)|$. Therefore, if we only focus on the request oblivious algorithms, we only lose a factor of 3 in comparison to the optimal online algorithm. The following lemma states a key structural lemma for an optimal request oblivious algorithm.

▶ **Lemma 8.** *For every request oblivious algorithm $\mathcal{B}$, there exists a randomized request oblivious algorithm $\mathcal{B}'$ with the same expected cost which is not only oblivious to the last request, but also oblivious to all requests that have come prior to this.*

Proof. For any given request oblivious online algorithm $\mathcal{B}$, we construct an online algorithm $\mathcal{B}'$ which is oblivious to all of the requests as follows: For an input $\langle B_1, B_2, \ldots, B_{i-1}\rangle$ of configurations and probability distributions $\langle P_1, P_2, \ldots, P_t\rangle$, draw a sequence of requests $\langle r_1, r_2, \ldots, r_i\rangle$ from $\langle P_1, P_2, \ldots, P_t\rangle$ conditioned on the constraint that $\mathcal{B}$ would generate configurations $\langle B_1, B_2, \ldots, B_{i-1}\rangle$ for requests $\langle r_1, r_2, \ldots, r_{i-1}\rangle$. Now, report the output of $\mathcal{B}$ for inputs $\langle B_1, B_2, \ldots, B_{i-1}\rangle$, $\langle r_1, r_2, \ldots, r_i\rangle$, and $\langle P_1, P_2, \ldots, P_t\rangle$.

We define the cost of step i of algorithm B' as $d(B_{i-1}, B_i) + 2d(B_i, r_i)$. Due to the construction of algorithm $\mathcal{B}'$, the expected cost of this algorithm at every step i for a random sequence of requests is equal to the expected cost of algorithm $\mathcal{B}$ for a random sequence of requests drawn from $\langle P_1, P_2, \ldots, P_t \rangle$. Therefore, the expected cost of both algorithms for a random sequence of requests are equal and thus $|\mathcal{B}(\mathcal{M}, \langle P_1, P_2, \ldots, P_t \rangle)| = |\mathcal{B}'(\mathcal{M}, \langle P_1, P_2, \ldots, P_t \rangle)|$. ◀

Lemma 8 states that there always exists an optimal randomized request oblivious online algorithm that returns the configurations regardless of the requests. We call such an algorithm *non-adaptive*. Since a non-adaptive algorithm is indifferent to the sequence of the requests, we can assume it always generates a sequence of configurations just based on the distributions. For an optimal of such algorithms, all such sequence of configurations should be optimal as well. Therefore, there always exists an optimal non-adaptive online algorithm which is deterministic. By Lemma 7 not only do we know the optimal request oblivious algorithm is at most 3-approximation, but also the same holds for the optimal non-adaptive algorithm.

▶ **Theorem 9.** *There exists a sequence of configurations $\langle B_0, B_1, \ldots, B_t \rangle$ such that an online algorithm which starts with B_0 and always returns configuration B_i upon arrival of request r_i has an opproximation factor of at most 3.*

4 Fractional Solutions

In this section we provide a fractional online algorithm for the k-server problem that can be implemented in polynomial time. Note that by Theorem 9 we know that there exist configurations $\langle \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_t \rangle$ such that the expected cost of a non-adaptive algorithm that always returns these configurations is at most 3 times the cost of an optimal online algorithm. Therefore, we write an integer program to find such configurations with the least expected cost. Next, we provide a relaxed LP of the integer program and show that every feasible solution of such LP corresponds to a fractional online algorithm for the stochastic k-server problem. Hence, solving such a linear program, that can be done in polynomial time, gives us a fractional online algorithm for the problem.

4.1 Linear Program

Recall that given t independent distributions $\langle P_1, \ldots, P_t \rangle$ for online stochastic k-server, an adaptive algorithm can be represented by $t+1$ configurations $\langle B_0, \ldots, B_t \rangle$. Upon the arrival of each request r_i, we move the servers from configuration B_{i-1} to B_i and then one server serves r_i and goes back to its position in B_i. The objective is to find the configurations such that the cost of moving to new configurations in addition to the expected cost of serving the requests is minimized. Therefore the problem can formulated in an offline manner. First we provide an integer program in order to find a vector of configurations with the least cost.

The decision variables of the program represent the configurations, the movement of servers from one configuration to another, and the way that each possible request is served. In particular, at each time step τ:

- For each node v there is a variable $b_{\tau,v} \in N$ denoting the number of servers on node v.
- For each pair of nodes u and v, there is a movement variable $f_{\tau,u,v} \in N$ denoting the number of servers going from u to v for the next round.
- For each node v and possible request node r, there is a variable $x_{\tau,v,r} \in \{0, 1\}$ denoting whether r is served by v or not.

In the following integer program, the first set of constraints ensures the number of servers on nodes at each time is updated correctly according to the movement variables. The second set of constraints ensures that each possible request is served by at least one server. The third set of constraints ensures that no possible request is served by an empty node. By the definition, the cost of a sequence of configurations $\langle B_0, \ldots, B_t \rangle$ is $\sum_{i=1}^{t} d(B_{i-1}, B_i) + 2\sum_{i=1}^{t} d(B_i, r_i)$. Thus the objective is to minimize the expression

$$\sum_{\tau} \sum_{u,v} f_{\tau,u,v} d(u,v) + 2 \sum_{\tau} \sum_{v} \sum_{r} x_{\tau,v,r} \Pr(z \sim P_\tau = r) d(v,r),$$

where $\Pr(z \sim P_\tau = r)$ denotes the probability that r is requested at time τ.

$$\begin{aligned}
&\text{min.} && \sum_{\tau} \sum_{u,v} f_{\tau,u,v} d(u,v) + 2 \sum_{\tau} \sum_{v} \sum_{r} x_{\tau,v,r} \Pr(z \sim P_\tau = r) d(v,r) \\
&\forall \tau, v && b_{\tau+1,v} = b_{\tau,v} + \sum_{u} f_{\tau,u,v} - \sum_{u} f_{\tau,v,u}. \\
&\forall \tau, u, v && \sum_{v} x_{\tau,v,r} \geq 1. \\
&\forall \tau, v, r && x_{\tau,v,r} \leq b_{\tau,v}. \\
&\forall \tau && \sum_{v} b_{\tau,v} \leq k. \\
&\forall \tau, v, r && x_{\tau,v,r} \in \{0,1\}. \\
&\forall \tau, u, v && f_{\tau,u,v} \in N. \\
&\forall \tau, v && b_{\tau,v} \in N.
\end{aligned}$$

Now we consider the following relaxation of the above integer program.

$$\begin{aligned}
&\text{min.} && \sum_{\tau} \sum_{u,v} f_{\tau,u,v} d(u,v) + 2 \sum_{\tau} \sum_{v} \sum_{r} x_{\tau,v,r} \Pr(z \sim P_\tau = r) d(v,r) \\
&\forall \tau, v && b_{\tau+1,v} = b_{\tau,v} + \sum_{u} f_{\tau,u,v} - \sum_{u} f_{\tau,v,u}. \\
&\forall \tau, u, v && \sum_{v} x_{\tau,v,r} \geq 1. \\
&\forall \tau, v, r && x_{\tau,v,r} \leq b_{\tau,v}. \\
&\forall \tau && \sum_{v} b_{\tau,v} \leq k.
\end{aligned}$$

5 Reduction from Integral k-server to Fractional k-server

In this section we show how we can obtain an integral algorithm for the stochastic k-server problem from a fractional algorithm. We first show that every fractional algorithm for the line metric can be modified to an integral algorithm with the same cost. Next, we study the problem on HST metrics; we give a rounding method that produces an integral algorithm from a fractional algorithm while losing a constant factor. Finally, we leverage the previously known embedding techniques to show every metric can be embedded into HST's with a distortion of at most $O(\log n)$. This will lead to a rounding method for obtaining an integral algorithm from every fractional algorithm on general metrics while losing a factor of at most $O(\log n)$. Combining this with the 3 approximation fractional algorithm that we provide

in Section 4, we achieve an $O(\log n)$ approximation algorithm for the stochastic k-server problem on general graphs.

5.1 Integrals Are as Strong as Fractionals On the Line

In this section we show every fractional algorithm on the line metric can be derandomized to an integral solution with the same expected cost. The rounding method is as follows: For every fractional configuration A, we provide an integral configuration $\mathrm{I}(A)$ such that (i) the distance of two configurations A_1 and A_2 is equal to the expected distance of two configurations $\mathrm{I}(A_1)$ and $\mathrm{I}(A_2)$. (ii) for every point x in the metric that A has a server mass of size at least 1 on x, there exists a server on point x in $\mathrm{I}(A)$.

Let for every point x in the metric, $A(v)$ denote the amount of server mass on node v of the line. For every fractional configuration B, we define a mass function $f_A : (0, k] \to V$ as follows. $f_A(x) = v_j$ if and only if j is the minimum integer such that $\sum_{i=1}^{j-1} A(i) < x$ and $\sum_{i=1}^{j} A(i) \geq x$. Intuitively, if one gathers the server mass by sweeping the line from left to right, $f_A(x)$ is the first position on which we have gathered x amount of server mass. The rounding algorithm is as follows:

- Pick a random real number r in the interval $[0, 1)$.
- $\mathrm{I}(A)$ contains k servers on positions $f_A(r)$, $f_A(r+1)$, $f_A(r+2)$, $\ldots$, $f_A(r+k-1)$.

Note that the rounding method uses the same r for all of the configurations. More precisely, we draw r from $[0, 1)$ at first and use this number to construct the integral configurations from fractional configurations. The following two lemmas show that both of the properties hold for the rounding algorithm we proposed.

▶ **Lemma 10.** *Let A be a fractional configuration and x be a point such that $A(x) \geq 1$. Then $\mathrm{I}(A)$ has a server on x.*

Proof. Due to the construction of our rounding method, for every two consecutive servers a and b in $\mathrm{I}(\mathcal{A})$, the total mass of servers after a and before b in the fractional solution is less than 1. Therefore, $\mathrm{I}(A)$ should put a server on point x, otherwise the total mass of servers in the fractional solution between the first server before x and the first server after x would be at least 1. ◀

The next lemma shows that the rounding preserves the distances between the configurations in expectation.

▶ **Lemma 11.** *Let A_1 and A_2 be two fractional configurations and $|A_1 - A_2|$ be their distance. The following holds for the distances of the configurations*

$$\mathbb{E}|\,\mathrm{I}(A_1) - \mathrm{I}(A_2)| = |A_1 - A_2|.$$

Proof. The key point behind the proof of this lemma is that the distance of two fractional configurations A_1 and A_2 can be formulated as follows

$$|A_1 - A_2| = \int_0^1 |\,\mathrm{I}_\omega(A_1) - \mathrm{I}_\omega(A_2)| d_\omega$$

where $\mathrm{I}_\omega(A)$ stands for an integral configurations which places the servers on points $f_A(\omega)$, $f_A(\omega+1)$, $f_A(\omega+2)$, $\ldots$, $f_A(\omega+k-1)$. Since at the beginning of the rounding method we draw r uniformly at random, the expected distance of the two rounded configurations is exactly equal to

$$\int_0^1 |\,\mathrm{I}_\omega(A_1) - \mathrm{I}_\omega(A_2)| d_\omega$$

which is equal to the distance of A_1 from A_2. ◀

▶ **Theorem 12.** *For any given fractional online algorithm $\mathcal{A}$ for the k-server problem on the line metric, there exists an online integral solution for the same problem with the same expected cost.*

5.2 Reduction for General Graphs

An HST is a undirected rooted tree in which every leaf represents a point in the metric and the distance of a pair of points in the metric is equal to the distance of the corresponding leaves in the tree. In an HST, weights of the edges are uniquely determined by the depth of the vertices they connect. More precisely, in a σ-HST the weight of an edges between a vertex v and its children is equal to σ^{h-d_v} where h stands for the height of the tree and d_v denotes the depth of vertex v.

Since HSTs are very well structured, designing algorithms on HSTs is relatively easier in comparison to a more complex metric. Therefore, a classic method for alleviating the complexity of the problems is to first embed the metrics into HSTs with a low distortion and then solve the problems on these trees.

Perhaps the most important property of the HSTs is the following:

▶ Observation 1. For every pair of leaves $u, v \in T$ of an HST, the distance of u and v is uniquely determined by the depth of their deepest common ancestor.

Note that, the higher the depth of the common ancestor is, the lower the distance of the leaves will be. Therefore, the closest leaves to a leaf v are the ones that share the most common ancestors with v. Bansal *et al.* propose a method for rounding every fractional solution of the k-server problem to an integral solution losing at most a constant factor [7].

▶ **Theorem 13** ([7])**.** *Let T be a σ-HST with n leaves, $\sigma > 5$, and let $A = \langle A_0, A_1, A_2, \ldots, A_t \rangle$ be a sequence of fractional configurations. There is an online procedure that maintains a sequence of randomized k-server configurations $S = \langle S_0, S_1, S_2, \ldots, S_t \rangle$ satisfying the following two properties:*

- *At any time i, the state S_i is consistent with the fractional state A_i.*
- *If the fractional state changes from x_{i-1} to x_i at time i, incurring a movement cost of c_i, then the state S_{i-1} can be modified to a state S_i while incurring a cost of $O(c_i)$ in expectation.*

Embedding general metrics into trees and in particular HSTs has been the subject of many studies. The seminal work of Fakcharoenphol *et al.* [20] has shown that any metric can be randomly embedded to σ-HSTs with distortion $O(\frac{\sigma \log n}{\log \sigma})$.

▶ **Theorem 14** ([20])**.** *There exists a probabilistic method to embed an arbitrary metric $\mathcal{M}$ into σ-HSTs with distortion $\frac{\sigma \log n}{\log \sigma}$.*

Therefore, to round a fractional solution on a general metric, we first embed it into 6-HSTs with a distortion of at most $O(\log n)$ and then round the solution while losing only a constant factor. This will give us an integral algorithm that has an expected cost of at most $O(\log n)$ times the optimal.

▶ **Theorem 15.** *For any given fractional online algorithm $\mathcal{A}$ for the k-server problem on an arbitrary metric, there exists an online integral solution for the same problem having a cost of no worse that $O(\log n)$ times the cost of $\mathcal{A}$ in expectation.*

Acknowledgment. We would like to thank Shi Li for having helpful discussions.

References

1. Melika Abolhasani, Soheil Ehsani, Hosein Esfandiari, MohammadTaghi Hajiaghayi, Robert Kleinberg, and Brendan Lucier. Beating $1 - 1/e$ for ordered prophets. *arXiv preprint arXiv:1704.05836*, 2017.
2. Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1):203–218, 2000.
3. Saeed Alaei, Mohammad T Hajiaghayi, Vahid Liaghat, Dan Pei, and Barna Saha. Adcell: Ad allocation in cellular networks. In *Algorithms–ESA 2011*, pages 311–322. Springer, 2011.
4. Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 18–35. ACM, 2012.
5. Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 11–25. Springer, 2013.
6. Susanne Albers, Lene M Favrholdt, and Oliver Giel. On paging with locality of reference. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 258–267. ACM, 2002.
7. Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 267–276. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.63.
8. Nikhil Bansal, Niv Buchbinder, and Joseph Seffi Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):19, 2012.
9. Yair Bartal, Marek Chrobak, and Lawrence L Larmore. A randomized algorithm for two servers on the line. *Information and Computation*, 158(1):53–69, 2000.
10. Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theoretical computer science*, 324(2):337–345, 2004.
11. Luca Becchetti. Modeling locality: A probabilistic analysis of lru and fwf. In *Algorithms–ESA 2004*, pages 98–109. Springer, 2004.
12. Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, 1999.
13. Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.
14. Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995.
15. Marek Chrobak, H Karloff, T Payne, and S Vishwnathan. New ressults on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.
16. Marek Chrobak and Lawrence L Larmore. An optimal on-line algorithm for k servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.
17. Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Online survivable network design and prophets. 2015.
18. Sina Dehghani, Ian A Kash, and Peter Key. Online stochastic scheduling and pricing the clouds. 2017.
19. Peter J Denning. The working set model for program behavior. *Communications of the ACM*, 26(1):43–48, 1983.
20. Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455. ACM, 2003.

21 Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

22 Amos Fiat and Manor Mendel. Truly online paging with locality of reference. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 326–335. IEEE, 1997.

23 Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.

24 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 942–951. Society for Industrial and Applied Mathematics, 2008.

25 Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, volume 7, pages 58–65, 2007.

26 Sandy Irani, Anna R Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, 1996.

27 Anna R Karlin, Steven J Phillips, and Prabhakar Raghavan. Markov paging. *SIAM Journal on Computing*, 30(3):906–922, 2000.

28 Howard Karloff, Yuval Rabani, and Yiftach Ravid. Lower bounds for randomized k-server and motion-planning algorithms. *SIAM Journal on Computing*, 23(2):293–312, 1994.

29 Elias Koutsoupias and Christos H Papadimitriou. On the k-server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983, 1995.

30 Ulrich Krengel, Louis Sucheston, et al. Semiamarts and finite values. *Bull. Amer. Math. Soc*, 83(4), 1977.

31 Mark S Manasse, Lyle A McGeoch, and Daniel D Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

32 Lyle A McGeoch and Daniel D Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(1-6):816–825, 1991.

33 Konstantinos Panagiotou and Alexander Souza. On adequate performance measures for paging. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 487–496. ACM, 2006.

34 Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

35 Duru Türkoglu. The k-server problem and fractional analysis, 2005. Master's Thesis, The University of Chicago. URL: `http://people.cs.uchicago.edu/~duru/papers/masters.pdf`.

36 Neal E Young. Bounding the diffuse adversary. In *SODA*, volume 98, pages 420–425, 1998.

Multiple Source Dual Fault Tolerant BFS Trees*

Manoj Gupta[1] and Shahbaz Khan†[2]

1 **IIT Gandhinagar, Gandhinagar, India**
gmanoj@iitgn.ac.in
2 **Department of CSE, IIT Kanpur, Kanpur, India**
shahbazk@cse.iitk.ac.in

Abstract

Let $G = (V, E)$ be a graph with n vertices and m edges, with a designated set of σ sources $S \subseteq V$. The *fault tolerant subgraph* for any graph problem maintains a sparse subgraph $H = (V, E')$ of G with $E' \subseteq E$, such that for any set F of k failures, the solution for the graph problem on $G \setminus F$ is maintained in its subgraph $H \setminus F$. We address the problem of maintaining a fault tolerant subgraph for computing *Breath First Search tree* (BFS) of the graph from a single source $s \in V$ (referred as k FT-BFS) or multiple sources $S \subseteq V$ (referred as k FT-MBFS). We simply refer to them as FT-BFS (or FT-MBFS) for $k = 1$, and dual FT-BFS (or dual FT-MBFS) for $k = 2$.

The problem of k FT-BFS was first studied by Parter and Peleg [ESA13]. They designed an algorithm to compute FT-BFS subgraph of size $O(n^{3/2})$. Further, they showed how their algorithm can be easily extended to FT-MBFS requiring $O(\sigma^{1/2}n^{3/2})$ space. They also presented matching lower bounds for these results. The result was later extended to solve dual FT-BFS by Parter [PODC15] requiring $O(n^{5/3})$ space, again with matching lower bounds. However, their result was limited to only edge failures in undirected graphs and involved very complex analysis. Moreover, their solution doesn't seems to be directly extendible for dual FT-MBFS problem.

We present a similar algorithm to solve dual FT-BFS problem with a much simpler analysis. Moreover, our algorithm also works for vertex failures and directed graphs, and can be easily extended to handle dual FT-MBFS problem, matching the lower bound of $O(\sigma^{1/3}n^{5/3})$ space described by Parter [PODC15]. The key difference in our approach is a much simpler classification of path interactions which formed the basis of the analysis by Parter [PODC15].

1998 ACM Subject Classification E.1 Graphs and Networks, G.2.2 Graph Algorithms, Network problems, Trees, G.4 Algorithm Design and Analysis, F.2.2 Computations on Discrete Structures

Keywords and phrases BFS, fault-tolerant, graph, algorithms, data-structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.127

1 Introduction

Graph networks are extensively used to study real world applications ranging from communication networks as internet and telephony, to supply chain networks, road networks etc. Every now and then, these networks are susceptible to failures of links and nodes, which drastically affects the performance of these applications. Hence, most algorithms developed for these applications are also studied in the *fault tolerant model*, which aims to provide solutions to the corresponding problem that are resilient to such failures. Since such failures of nodes or links in the network though unpredictable are rare and are often readily repaired, the applications generally address the scenarios expecting the number of simultaneous faults to

* The full version of the paper can be found in [11].
† This research work was supported by Google India under the Google India PhD Fellowship Award.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 127; pp. 127:1–127:15


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

be much smaller than the size of the network. This aspect is often modeled by bounding such failures using some parameter k (typically $k << n$), and studying fault tolerant structures resilient to upto k failures.

Among the different approaches to develop fault tolerance in a structure, we use the approach of computing a *fault tolerant subgraph* described as follows. For a given graph $G = (V, E)$, the fault tolerant subgraph for any graph problem maintains a sparse subgraph $H = (V, E')$ of G having $E' \subseteq E$, such that for any set of edge (or vertex) failures $F \subseteq E$ (or $F \subseteq V$), the solution for the graph problem on $G' = (V, E \setminus F)$ (or $G' = (V \setminus F, E)$) is maintained in its subgraph $H' = (V, E \setminus F)$ (or $H' = (V \setminus F, E)$). We shall henceforth abuse the notation to denote the graphs after such a set of failures F as $G \setminus F$ and $H \setminus F$ respectively. A standard motivation for this approach is a communication network where each link corresponds to a communication channel [16], where the system designer is required to purchase or lease the channels to be used by the application. Hence, the aim is to acquire a minimal set of these channels (the subgraph H of G) for successfully performing the application with resilience of upto k faults. Fault tolerant subgraphs are also developed for other graph problems maintaining reachability [13, 2, 3], strong-connectivity [3] and approximate shortest paths from a single source [12, 17, 5] and all sources [7, 9, 6, 14, 4].

Breadth First Search (BFS) is a fundamental technique for graph traversal. From any given source $s \in V$, BFS produces a rooted spanning tree in $O(m + n)$ time. For an unweighted graph, the BFS tree from a source s is also the shortest path tree from s because it preserves the shortest path from s to every vertex $v \in V$ that is reachable from s. We are thus interested to maintain fault tolerant subgraphs for computing BFS trees from a single source (referred as k FT-BFS) and multiple sources k FT-MBFS described as follows.

▶ **Definition 1** (k FT-BFS)**.** Given a graph $G = (V, E)$ with a designated source $s \in V$, build a subgraph $H = (V, E')$ with $E' \subseteq E$, such that after any set F of k failures in G, the BFS tree from s in $H \setminus F$ is a valid BFS tree from s in $G \setminus F$.

▶ **Definition 2** (k FT-MBFS)**.** Given a graph $G = (V, E)$ with a designated set of sources $S \subseteq V$, build a subgraph $H = (V, E')$ with $E' \subseteq E$, such that after any set F of k failures in G, for each $s \in S$ the BFS tree from s in $H \setminus F$ is a valid BFS tree from s in $G \setminus F$.

For convenience of notation, for $k = 1$ and $k = 2$ we refer to these problems as FT-BFS (or FT-MBFS) and dual FT-BFS (or dual FT-MBFS). The problems of k FT-BFS (and k FT-MBFS) were first studied by Parter and Peleg [16] for a single failure. They designed an algorithm to compute FT-BFS requiring $O(n^{3/2})$ space. Further, they showed their result can be easily extended to FT-MBFS requiring $O(\sigma^{1/2}n^{3/2})$ space. Moreover, their upper bounds were complemented by matching lower bounds for both their results. This result was later extended to address dual FT-BFS by Parter [15] requiring $O(n^{5/3})$ space. However, the application of this result was limited to only edge failures in undirected graphs. Though the analysis of their result was significantly complex, it paved a way for developing the theory studying the interaction of replacement paths after a single edge failure, their classification and corresponding properties. Further, they also generalized the lower bound for k FT-MBFS to $\Omega(\sigma^{\frac{1}{k+1}}n^{2-\frac{1}{k+1}})$ which matches their solution for dual FT-BFS. They also stated extensions of their result to dual FT-MBFS (or k FT-BFS) as an open problem.

The difference in complexity of dual FT-BFS over FT-BFS also reinforces the idea that extending such results from one failure to two failures (and beyond) requires a significantly more advanced analysis. As described by Parter [15], for the problem of maintaining shortest paths *"a sharp qualitative and quantitative difference"* has been widely noted while handling a single failure and multiple failures. For the problem of maintaining fault tolerant distance

oracles, despite a simple and elegant algorithm for a single edge failure [8], the solution for two edge failures [10] is significantly complex. In fact, the authors [10] themselves mention that extending their approach beyond 2 edge failure would be infeasible due to numerous case analysis involved, requiring a fundamentally different approach. This key difference is also visible when we compare other problems, as bi-connectivity with tri-connectivity, single fault tolerant reachability [13, 2] with dual fault tolerant reachability [3], etc. Hence, simplifying the analysis of dual FT-BFS (and hence dual FT-MBFS) structures seem to be an essential building block for further developments of the problem for multiple failures.

1.1 Our Contributions

We design optimal algorithms for constructing dual FT-BFS and dual FT-MBFS structures. In principle, the core algorithm of our construction for dual FT-BFS is same as the one given by Parter [15], with a much simpler and more powerful analysis. As a result, our algorithm also works for vertex failures and directed graphs. Also, our dual FT-BFS structure can also be easily extended to handle dual FT-MBFS (as in case of FT-BFS [16]), which matches the lower bound described by Parter [15]. Thus, we optimally solve two open problems (dual FT-BFS for directed graphs and dual FT-MBFS for any graphs) as follows.

▶ **Theorem 3** (Optimal dual FT-BFS). *Given any graph $G = (V, E)$ having n vertices and m edges, with a designated source $s \in V$, there is a polynomial time constructable dual FT-BFS subgraph H having $O(n^{5/3})$ edges.*

▶ **Theorem 4** (Optimal dual FT-MBFS). *Given any graph $G = (V, E)$ having n vertices and m edges, with a designated set of σ sources $S \subseteq V$, there is a polynomial time constructable dual FT-MBFS subgraph H having $O(\sigma^{1/3} n^{5/3})$ edges.*

Our analysis is performed using simple techniques based on *counting arguments*. We classify a set of shortest paths as *standard* paths and prove the properties of *disjointness* and *convergence* for a designated suffix of such paths. The extension to directed graphs additionally uses the notion of *segmentable* paths (similar notion of *regions* was used in [15]) for every set of *converging* shortest paths, and establishes several interesting properties for them. These properties and analysis techniques might be of independent interest in the theory of shortest paths.

1.2 Related Work

As described earlier BFS is strongly related to shortest paths. Demetrescu et al. [8] showed that there exist weighted directed graphs, for which a fault tolerant subgraph requires $\Theta(m)$ edges for maintaining shortest paths even from a single source after a vertex failure. Hence, they designed a data-structure of size $\tilde{O}(n^2)$ [1] that reports all pairs shortest distances after a vertex failure in $O(1)$ time. Duan and Pettie [10] extended this result to two failures requiring nearly same (upto $poly \log n$ factors) size and reporting time.

Other related problems include fault tolerant DFS and fault tolerant reachability. Baswana et al. [1] presented a $\tilde{O}(m)$ sized fault tolerant data structure that reports the DFS tree of an undirected graph after k faults in $\tilde{O}(nk)$ time. For single source reachability, Baswana et al. [3] presented an algorithm for computing fault tolerant reachability subgraphs for k faults using $O(2^k n)$ edges. This result was also shown to be optimal upto constant factors.

[1] $\tilde{O}(\cdot)$ notation hides poly-log(n) factors

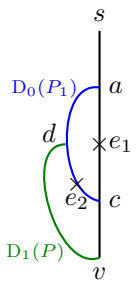


Figure 1 Showing P_0 (in black), $\mathrm{D}_0(P_1)$ (in blue) and $\mathrm{D}_1(P)$ (in green). Here $P_1 = P_0[s,a] \cup \mathrm{D}_0(P_1) \cup P_0[c,v]$ and $P = P_0[s,a] \cup \mathrm{D}_0(P_1)[a,d] \cup \mathrm{D}_1(P)$.

1.3 Outline of the paper

We now present a brief outline of our paper. In Section 2, we present the basic notations that shall be used throughout the paper, which shall be followed by a brief overview of our approach and analysis in Section 3. In Section 4, we shall first begin with the description of our algorithm for dual FT-BFS and the properties of the shortest paths found using it, which shall be followed by the formal analysis. We then present our algorithm for dual FT-MBFS and its analysis, drawing similarities with solution of dual FT-BFS. Finally, we present the concluding remarks for our paper in Section 6. Due to page constraints some proofs have been omitted and deferred to the full paper [11]. For the sake of simplicity, we only describe our algorithm and analysis for edge failures. However, the same analysis can also be used to handle vertex failures.

2 Preliminaries

Given a graph $G = (V, E)$ with n vertices and m edges with a set of designated source $s \in S$. The following notations shall be used throughout the paper.

- $P, \mathcal{P}$: A path is denoted by P, where $Source(P)$ and $Dest(P)$ represents the source and destination of path P. In most parts of the paper, $Source(P) = s$ and $Dest(P) = v$. A set of paths is denoted by $\mathcal{P}$. Generally, we assume a path from s to v starts from the top (s) and ends at bottom (v). For two paths P', P'', we say P' leaves *earlier/higher* (or *later/lower*) than P'' from P, if P' leaves P closer to s (or closer to v) than P''.
- $\mathrm{F}(P)$: For the shortest path P from $Source(P)$ to $Dest(P)$ after a set of edge failures, this set of failed edges is denoted by $\mathrm{F}(P) = \{e_1, e_2, \ldots, e_k\}$ (say), where e_i denotes the i^{th} edge in the sequence. Similarly for some path P', e'_i denotes the i^{th} edge in the sequence.
- P_i: If $\mathrm{F}(P) = \{e_1, e_2, \ldots, e_k\}$, then P_i is the shortest path avoiding the first i edge of $\mathrm{F}(P)$, i.e., $\mathrm{F}(P_i) = \{e_1, e_2, \ldots, e_i\}$, where $0 \leq i < k$. Again, for most parts of the paper, P_0 denotes the shortest path from s to v in G.
- $\mathrm{D}_i(P)$: If $|\mathrm{F}(P)| = k$, the detour path of P from P_i, $\mathrm{D}_i(P) = P \setminus \{\cup_{j=0}^{i} P_j\}$ [2], where $1 \leq i < k-1$. For dual case, $D_0(P)$ is the detour of P from P_0, $\mathrm{D}_1(P)$ is the detour of P from P_1, and $\mathrm{D}_0(P_1)$ is the detour of P_1 from P_0 (See Figure 1).
- $LastE(P)$: The last edge of a path P.
- $P[x, y]$: The sub-path of P starting from x to y, where $x, y \in P$.

[2] This construction may give a set of disjoint subpaths of P instead of a single subpath. However, in most cases this path will be a single subpath, else we assume $\mathrm{D}_i(P)$ to be the last such subpath on P.

We define the property of *convergence* of a set of paths $\mathcal{P}$ as follows. The paths in $\mathcal{P}$ are said to be *converging* if on intersection of any two paths $P, P' \in \mathcal{P}$, both P and P' merge and do not diverge till the end of the paths.

3 Overview

For analyzing the size of dual FT-BFS subgraph, i.e., the number of edges in shortest paths from the source s to each vertex $v \in V$ after any two failures, it suffices to count only the last edge of every such path P, for each $v \in V$ [16, 15]. The novelty of our approach is the classification of such paths based on interaction of corresponding P_1 and P_0, whereas Parter [15] studied the different interactions of P_1 and P'_1, for two such paths P and P'.

We primarily use the disjointness of a designated suffix of such a path P (referred as $LastLeg(P)$) with *counting* arguments to bound the number of such paths. To achieve this, we classify some of these paths as *standard paths* based on the interactions of corresponding P_1 and P_0. The number of *non-standard* paths can be easily bound using simple *counting* arguments. The set of *standard* paths exhibit several interesting properties including convergence of corresponding paths $\mathrm{D}_0(P_1)$. We further classify the *standard* paths into *long standard* paths and *short standard* paths, each bounded separately using relatively harder techniques. For sake of easier presentation we first bound the number of *short standard* paths only for undirected graphs, with extension to directed graphs requiring an additional notion of *segmentable* paths. The only difference in the analysis of dual FT-MBFS is the definition of *standard* paths and dealing with interaction of P_1 with P'_0 corresponding to other sources.

4 Dual FT-BFS

We shall now describe our algorithm to compute sparse dual FT-BFS subgraph H from a source $s \in V$. For every vertex $v \in V$, our algorithm computes the shortest paths from s to v avoiding upto two failures and adds the last edge of each such path to the adjacency list of vertex v. Note that repeating the procedure for each vertex on such a path adds the entire path to H [16, 15].

Our algorithm starts by adding the shortest path between s and v, i.e., P_0. It then processes single edge failures on P_0. We then find the replacement path P for all two edge failures $\{e_1, e_2\}$ such that $e_1 \in P_0$ and $e_2 \in P_1$. Further, in case $e_2 \in P_0 \cap P_1$ then e_1 is higher than e_2 on P_0.

However, we want to process all the failures in some particular order. This ordering plays a crucial role in the analysis. To this end, we define this ordering π as follows. The first failure in π is $F = \emptyset$, which adds P_0. The ordering π then contains single edge failures of type $F = \{e\}$ (where $e \in P_0$), ordered by their decreasing distance from s on P_0. Finally, we order the remaining failures as follows: for any two failures $F = \{e_1, e_2\}$ and $F' = \{e'_1, e'_2\}$ (with corresponding replacement paths P and P'), $F \prec_\pi F'$ if either (1) e_1 is farther than e'_1 from s on P_0, or (2) $e_1 = e'_1$ and e_2 is farther than e'_2 from s on P_1 (note that $P_1 = P'_1$ in this case). If $F \prec_\pi F'$, F is said to be *lower* than F' in π.

For any failure of $F = \{e_1, \cdots, e_k\}$, we define the *preferred* shortest path avoiding F. Our preferred shortest path will be a path of shortest length avoiding F. However, there can be multiple such paths of same length. We use following rules to choose a unique preferred path.

Procedure Dual-FT-BFS(s, v, π): Augments the dual FT-BFS subgraph H, such that for BFS tree of G rooted at s after any two edge failures in G, the incoming edges to v are preserved in H.

```
1 foreach Failure F, where 0 ≤ |F| ≤ 2, ordered from lower to higher in π do
2     P ← Preferred path from s to v in G avoiding F;
3     if LastE(P) ∉ H then
4         Assign P for failure of F;
5         Add LastE(P) to H;
6     end
7 end
```

▶ **Definition 5.** Path P is **preferred** for failure of $\{e_1, \cdots, e_k\}$ where each $e_i \in P_{i-1}$, if

1. For each i, P leaves P_{i-1} before e_i exactly once.
2. For any other P' also avoiding $\{e_1, \cdots, e_i\}$, we have either (i) $|P| < |P'|$, (ii) $|P| = |P'|$, and for some $0 \leq i \leq k$, both P and P' leaves each of $P_0, ..., P_{i-1}$ at the same vertex, but P leaves P_i earlier than P', (iii) P is lexicographically smaller [3] than P'.

Intuitively, out of all the shortest paths avoiding F (say for $|F| = 2$), the preferred path leaves the path P_0 and/or P_1 as early as possible. In order to avoid the preferred path leaving P_0 (or P_1) multiple times just to achieve an earlier point of divergence from P_0 (or P_1), the first condition is imposed. The last condition in (2) is just to break ties between two paths that are of same length and leave P_0 and P_1 at the same vertex.

Finally, in order to add the preferred shortest path P avoiding a failure F, our algorithm simply adds $LastE(P)$ to H, which suffices to add the entire path as described earlier. Moreover, we also assign the corresponding P to the failure F if it was the first failure to add this edge in H. As a result, if P and P' are two preferred paths avoiding F and F' respectively where $LastE(P) = LastE(P')$, then if $F \prec_\pi F'$, only the path P shall be assigned to F. Refer to Procedure Dual-FT-BFS for the pseudocode of our algorithm.

In order to calculate the size of H, it is sufficient to analyze the number of different last edges added on each $v \in V$ in H. Let the set of all paths from s to v avoiding failures $F \subseteq E$ (where $|F| \leq 2$) be $\mathcal{P}_v$. We thus define the paths that will be counted for establishing the space bound as follows.

▶ **Definition 6.** The path $P \in \mathcal{P}_v$ is called *contributing* if while processing $\mathrm{F}(P)$, $LastE(P) \notin H$, i.e., P adds a new edge adjacent to v in H.

In order to count the number of contributing paths to a vertex v, we only need to consider its interactions with other contributing paths in $\mathcal{P}_v$. This is because, if any other path $P \in \mathcal{P}_x$ passes through v using some new edge, so does the corresponding $P' \in \mathcal{P}_v$ with $\mathrm{F}(P) = \mathrm{F}(P')$. Thus, to analyze the size of H, it suffices to look at last edges of the contributing paths in $\mathcal{P}_v$ for each vertex v separately.

[3] Let P and P' first diverge from each other to $x \in P$ and $x' \in P'$ respectively, i.e., $P[s, x] \setminus \{x\} = P'[s, x'] \setminus \{x'\}$. If the index of x is lower than that of x' then P is said to be *lexicographically smaller* than P'.

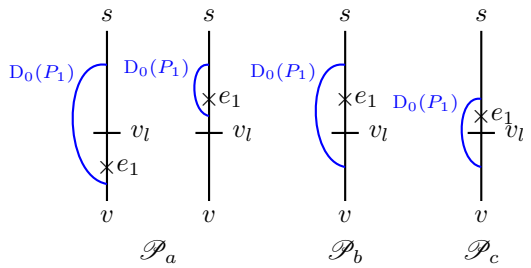


Figure 2 Classification of contributing paths: $\mathscr{P}_a$: Non-Standard Paths, $\mathscr{P}_b$: Long Standard Paths and $\mathscr{P}_c$: Short Standard Paths.

4.1 Properties of contributing paths

Parter [15] presented a simple proof bounding the number of contributing paths avoiding multiple failures on P_0 to $O(\sqrt{n})$ for each vertex v (an alternate proof using *counting arguments* is presented in the full paper [11]). Hence, excluding these paths, every contributing path satisfies the following properties (see full paper [11] for proofs).

▶ **Lemma 7.** *Excluding $O(\sqrt{n})$ paths, each contributing path P from s to v avoiding $\{e_1, e_2\}$ satisfies following properties*

$\mathbb{P}_1$: *$e_1 \in P_0$ and $e_2 \in \mathrm{D}_0(P_1)$.*

$\mathbb{P}_2$: *Except at v, $\mathrm{D}_0(P)$ does not intersect with P_0 and $\mathrm{D}_1(P)$ does not intersect with P_1, after diverging from P_0 and P_1 respectively.*

$\mathbb{P}_3$: *For any path P' which avoids $\{e_1, e_2\}$, P is the preferred path over P'.*

$\mathbb{P}_4$: *If P also avoids some failure F' where $F' \prec_\pi F$, then there exist another path P' which is the preferred path for F' over P, where P' does not avoid F.*

4.2 Space Analysis

As described earlier, in order to bound the size of dual FT-BFS subgraph to $O(n^{5/3})$, it suffices to bound the number of *contributing* paths from s to each vertex $v \in V$ avoiding two edge failures to $O(n^{2/3})$. Further, using $\mathbb{P}_1$ we are only concerned with a contributing path P if $e_1 \in P_0$ and $e_2 \in \mathrm{D}_0(P_1)$.

We first divide the path P_0 into two parts as follows. Let $v_l \in P_0$ be the vertex such that $|P_0[v_l, v]| = n^{1/3}$. We define $P_{high} = P_0[s, v_l]$ and $P_{low} = P_0[v_l, v]$. If $|P_0| < n^{1/3}$, we assume $v_l = s$ where $P_{high} = \phi$. We shall now define the *standard paths* as follows.

▶ **Definition 8** (Standard Paths)**.** A contributing path P is called a *standard path* if (a) $e_1 \in P_{high}$, and (b) $\mathrm{D}_0(P_1)$ merges with P_0 on P_{low}, i.e., $Dest(\mathrm{D}_0(P_1)) \in P_{low}$.

We can thus classify the contributing paths into following three types (see Figure 2):

$\mathscr{P}_a$: Non-standard paths.

$\mathscr{P}_b$: Long standard paths, i.e., standard paths with $|\mathrm{D}_0(P_1)| \geq n^{2/3}$.

$\mathscr{P}_c$: Short standard paths, i.e., standard paths with $|\mathrm{D}_0(P_1)| < n^{2/3}$.

Clearly, the sets $\mathscr{P}_a$, $\mathscr{P}_b$ and $\mathscr{P}_c$ are mutually disjoint and collectively exhaustive. Further, we define a set $\mathscr{P}_{1x}$ (for $x = a, b$ and c), where for each $P \in \mathscr{P}_x$, we add the corresponding P_1 to $\mathscr{P}_{1x}$. In addition, we identify the disjoint suffix of a path P as follows (see Figure 3).

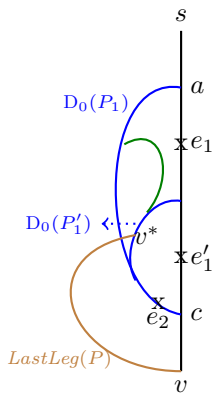


Figure 3 P avoids $\{e_1, e_2\}$. Its detour $D_1(P)$ (shown in blue) last intersects $LastPath(P) = P_1'$. P diverges from P_1' at v^*, i.e., $LastLeg(P) = P[v^*, v]$ (shown in brown).

▶ **Definition 9.** For each $P \in \mathscr{P}_x$, for $x = a, b$ or c, we define the following

1. $LastPath(P)$: The path in $\mathscr{P}_{1x}$ that intersects last with P. If P diverges from P_0 and does not intersect any path in $\mathscr{P}_{1x}$, we set $LastPath(P) = P_0$.
2. $LastLeg(P)$: The part of P after diverging from $LastPath(P)$, i.e., $P[v^*, v]$, where v^* is the last vertex of P on $P \cap LastPath(P)$.

The suffix $LastLeg(P)$ of a contributing path P satisfies the following properties (see full paper [11] for proofs).

▶ **Lemma 10.** *For every set $\mathscr{P}_x$ (for $x = a, b$ or c), we have the following.*

(a) *For any $P, P' \in \mathscr{P}_x$, $LastLeg(P)$ and $LastLeg(P')$ are disjoint (except at v), i.e., $LastLeg(P) \cap LastLeg(P') = \{v\}$. Further, each P, P' starts from a distinct vertex on $\mathscr{P}_{1x}$.*

(b) *Number of paths $P \in \mathscr{P}_x$ with $|LastLeg(P)| > n^{1/3}$ or $LastPath(P) = P_0$, is $O(n^{2/3})$.*

Remark: Lemma 102 claims that $LastLeg(P)$ is disjoint from other $LastLeg(P')$, where $P \in \mathscr{P}_x$ and $P' \in \mathscr{P}_{x'}$ only when $x = x'$. However, in case $x \neq x'$ they can intersect and our proof does not require their disjointness.

Equipped with these properties we can easily analyze the number of *non-standard paths* ($\mathscr{P}_a$) and *standard paths* ($\mathscr{P}_b$ and $\mathscr{P}_c$) in the following sections.

4.2.1 Analyzing non-standard paths $\mathscr{P}_a$

Using Lemma 102, we know that the number of $P \in \mathscr{P}_a$ with $|LastLeg(P)| > n^{1/3}$ or $LastPath(P) = P_0$ is $O(n^{2/3})$. We now focus on the case when $|LastLeg(P)| \leq n^{1/3}$ and $LastPath(P) \in \mathscr{P}_{1a}$. For any path P, let $v^* = Source(LastLeg(P))$. Since $LastLeg(P)$ is a detour from $LastPath(P)[v^*, v]$ avoiding the entire P_0 (using $\mathbb{P}_2$), we have $|LastPath(P)[v^*, v]| \leq |LastLeg(P)| \leq n^{1/3}$. By definition, a contributing path P is *non-standard* if either (a) $e_1 \in P_{low}$, or (b) $D_0(P_1)$ merges with P_0 on P_{high}, i.e., $Dest(D_0(P_1)) \in P_{high}$. Hence, for every P, $LastPath(P)$ would correspond to one of the two cases (a) or (b). Case (b) is clearly not be applicable here because $|LastPath(P)[v^*, v]| \geq |P_{low}| = n^{1/3}$ (since $Dest(LastPath(P)) \in P_{high}$). For case (a), on each $LastPath(P) \in \mathscr{P}_{1a}$, v^* can be one of $n^{1/3}$ vertices of $LastPath(P)$ closest to v. Further, since $e_1 \in P_{low}$, there are only $n^{1/3}$ such paths in $\mathscr{P}_{1a}$ because each such path corresponds to failure of unique edge in P_{low}. Thus,

there are only $n^{1/3} \times n^{1/3} = n^{2/3}$ different vertices v^* limiting the number of $P \in \mathscr{P}_a$ with $|LastLeg(P)| \leq n^{1/3}$ to $O(n^{2/3})$ (using Lemma 101).

Properties of standard paths ($\mathscr{P}_b$ or $\mathscr{P}_c$)

We shall now prove two important properties of standard paths (see full paper [11] for proofs). The first result states that if $\mathrm{D}_0(P_1)$ and $\mathrm{D}_0(P_1')$ intersect, where $P, P' \in \mathscr{P}_{1b} \cup \mathscr{P}_{1c}$, then they cannot diverge. The second result states that the length of paths in $\mathscr{P}_b \cup \mathscr{P}_c$ are different. A similar result was proved by Parter[15].

▶ **Lemma 11.** *For the set of contributing* standard *paths, we have the following properties.*

(a) *The set of paths* $\{\mathrm{D}_0(P_1) | P_1 \in \mathscr{P}_{1b} \cup \mathscr{P}_{1c}\}$*, is converging.*

(b) *(Parter [15]) For any two paths* $P, P' \in \mathscr{P}_b \cup \mathscr{P}_c$*,* $|P| \neq |P'|$*.*

4.2.2 Analyzing long standard paths $\mathscr{P}_b$

We first prove a generic technique to bound the number of contributing paths P if the set of corresponding paths P_1 is converging and each P_1 sufficiently long.

▶ **Theorem 12.** *Given a set* $\mathcal{P}$ *of converging paths satisfying Lemma 101, where for each* $P_1 \in \mathcal{P}$ *we have* $|P_1| \geq \alpha^2$ *(where* $\alpha \geq 1$*), the number of contributing paths* P *having* $P_1 \in \mathcal{P}$ *is* $O(n/\alpha)$*.*

Proof. Recall the definition of $LastPath(P)$, here we shall define $LastPath(P)$ (and hence $LastLeg(P)$) corresponding to paths in $\mathcal{P}$ (rather than $\mathscr{P}_{1x}$ in Definition 9). Using Lemma 101, if $|LastLeg(P)| \geq \alpha$, then P can be associated with α unique vertices of $LastLeg(P)$. This limits the total number of such paths to $O(n/\alpha)$. Hence, we assume that $LastLeg(P)) \leq \alpha$.

For each path $P_1 \in \mathcal{P}$, let $v_l = Dest(P_1)$. Similarly, for each such P, let the last intersection vertex of $LastLeg(P)$ and $LastPath(P)$ be v^*. Using Lemma 101, we know that for each such contributing path P, its corresponding $LastLeg(P)$ starts from a distinct vertex of $\mathcal{P}$. Since $LastLeg(P)$ is a detour from $LastPath(P)[v^*, v_l]$ avoiding the entire P_1 (using $\mathbb{P}_2$), we have $|LastLeg(P)| \geq |LastPath(P)[v^*, v_l]|$. Since $|LastLeg(P)| \leq \alpha$, v^* can be one of α vertices of $LastPath(P)$ closest to v_l.

We shall associate each such vertex v^* on $LastPath(P) \in \mathcal{P}$ uniquely with α vertices of $LastPath(P)$, for all $LastPath(P) \in \mathcal{P}$, as follows. Let the vertices of some $LastPath(P)$ be $v_1, ..., v_k$ where v_1 is the closest vertex to v_l. For each v_i, $i = 1, ..., \alpha$, we associate the vertices $v_{(i-1)\alpha}, ..., v_{i\alpha}$. Since $|LastPath(P)| \geq \alpha^2$ (by definition of $\mathcal{P}$) and $i \in [1, \alpha]$ such an association can be made. Now, in order to prove that such an association is unique, i.e., a vertex x is not associated with two different vertices v_1^*, v_2^* of $\mathcal{P}$, we exploit the convergence of $\mathcal{P}$ as follows. Clearly if $x \in P_1$ for a unique path $P_1 \in \mathcal{P}$, there is a unique $v_1^* \in \mathcal{P}$ to which it is associated. However, if $x \in P_1$ and $x \in P_1'$ for any two paths $P_1, P_1' \in \mathcal{P}$, then P_1 and P_1' will not diverge after intersection (by convergence of $\mathcal{P}$). This implies $P_1[x, v_l] = P_1'[x, v_l']$. Thus, the corresponding $v_1^* \in P_1$ and $v_2^* \in P_1'$ would also be same as by definition $v_1^* \in P_1[x, v_l]$. Hence, for every P emerging from v^* with $|LastPath(P)[v^*, v_l]| \leq \alpha$, the corresponding v^* can be uniquely associated with at least α vertices of $\mathcal{P}$. This limits the total number of such paths to $O(n/\alpha)$ proving the theorem. ◀

Using Lemma 111 and by definition of long standard paths $\mathscr{P}_b$, Theorem 12 is applicable for the set $\mathrm{D}_0(P_1)$ for $P_1 \in \mathscr{P}_{1b}$ and $\alpha = n^{1/3}$ limiting the number of paths in $\mathscr{P}_b$ to $O(n^{2/3})$.

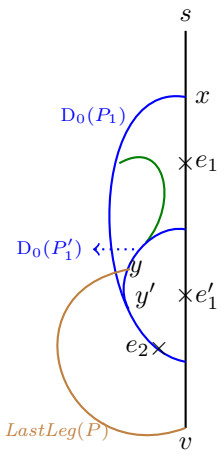


Figure 4 Let P_1' be $LastPath(P)$. Then the path $P_0[s,x] \cup P_1[x,y'] \cup P_1'[y',y] \cup LastLeg(P)$ is a valid path avoiding $\{e_1, e_2\}$.

4.2.3 Analyzing short standard paths $\mathscr{P}_c$

To highlight the simplicity of our approach, we only analyze the paths in $\mathscr{P}_c$ for undirected graphs here. For extension of this proof to handle directed graphs we use the theory of *segmentable paths* (refer to full paper [11] for details).

Using Lemma 102, we know that the number of $P \in \mathscr{P}_c$ with $|LastLeg(P)| > n^{1/3}$ or $LastPath(P) = P_0$ is $O(n^{2/3})$. We now focus on the case when $|LastLeg(P)| \leq n^{1/3}$ and $LastPath(P) \in \mathscr{P}_{1c}$. Any such contributing path P can be divided into two parts (see Figure 4), (a) $P[s,y]$, where $y = Source(LastLeg(P))$, and (b) $P[y,v] = LastLeg(P)$. We will now find an alternate path for $P[s,y]$, which will help us in bounding its length. Since P is a contributing path, it diverges from $LastPath(P)$ which requires either e_1 or e_2 to be on $LastPath(P)[y,v]$. By definition of *standard paths*, we have $D_0(LastPath(P))$ terminates on P_0 only on P_{low}, whereas $e_1 \notin P_{low}$ ensuring that $e_1 \notin LastPath(P)$. Thus, $e_2 \in LastPath(P)[y,v]$ and hence it intersects with P_1 as $e_2 \in P_1$. Using Lemma 111, we can thus say that $LastPath(P)$ and P_1 merge at some vertex say y', where $e_2 \in LastPath(P)[y',v] = P_1[y',v]$ (see Figure 4). We have an alternate path for $P[s,y]$ avoiding $F(P)$ formed by $P_1[s,y'] \cup LastPath(P)[y',y]$. Let $x = Source(D_0(P_1))$. Since $P[s,v]$ is the shortest path avoiding $F(P)$ we have

$$\begin{aligned}
|P| &= |P[s,y]| + |P[y,v]| \\
&= |P_1[s,y]| + |P[y,v]| \\
&\leq |P_1[s,y']| + |LastPath(P)[y',y]|) + |LastLeg(P)[y,v]| \\
&= (|P_1[s,x]| + |P_1[x,y']|) + |LastPath(P)[y',y]| + |LastLeg(P)[y,v]| \\
&\leq |P_0| + |D_0(P_1)| + |D_0(LastPath(P))| + |LastLeg(P)| \\
&\leq |P_0| + n^{2/3} + n^{2/3} + n^{1/3} \qquad \text{(by definition of } \mathscr{P}_c\text{)}
\end{aligned}$$

Now, using Lemma 112, we know that for any $P, P' \in \mathscr{P}_c$ we have $|P| \neq |P'|$. We thus arrange the paths in $\mathscr{P}_c$ (except the ones in Lemma 102) in the increasing order of sizes, where i^{th} such path has the length $\geq |P_0| + i$ (as all paths at least as long as P_0). Since for any such $P \in \mathscr{P}_c$ we also have $|P| \leq |P_0| + 3n^{2/3}$ (described above), clearly the number of paths in $\mathscr{P}_c$ are $O(n^{2/3})$ (for i upto $3n^{2/3}$).

Procedure Dual-FT-MBFS(S,v,π): Augments the dual FT-MBFS subgraph H, such that for BFS tree of G rooted at each $s \in S$ after any two edge failures in G, the incoming edges to v are preserved in H.

1 **foreach** $s \in S$ **do** Dual-FT-BFS(s,v,$\pi_s(0)$) ;
2 **foreach** $s \in S$ **do** Dual-FT-BFS(s,v,$\pi_s(1)$) ;
3 **foreach** $s \in S$ **do** Dual-FT-BFS(s,v,$\pi_s(2)$) ;

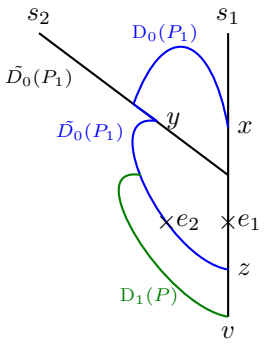


Figure 5 Shortest path avoiding $\{e_1, e_2\}$ is P. $D_1(P)$ last intersects $\tilde{P}_0(P_1) = P_0(s_2, v)$. P_1 diverges from $\tilde{P}_0(P_1)$ at y, i.e., $\tilde{D}_0(P_1) = P_1[y, z]$ (shown in blue). P also diverges from $\tilde{P}_0(P)$ at y, i.e., $\tilde{D}_0(P) = P[y, v]$.

This completes the proof of our dual FT-BFS result in Theorem 3.

5 Extension to dual FT-MBFS

In this section we shall extend our analysis of the previous section to handle σ sources using total $O(\sigma^{1/3}n^{5/3})$ space. We follow the approach similar to the case for single source. Let S be the set of sources, where $|S| = \sigma$. Given a source s, let $\pi_s \subseteq \pi$ denote the ordering of edge failure of size upto 2. Let $\pi_s(0), \pi_s(1)$ and $\pi_s(2)$ be the subset of π_s of size 0, 1 and 2 respectively. Our algorithm for finding dual FT-MBFS mimics the single source case.

The first for loop in the above procedure finds shortest path from each source to v. We shall refer to the set of the shortest paths from each source to v for different $s \in S$ as $\mathscr{P}_0$. We then move on to find the shortest path from each source to v avoiding one edge failure and two failures respectively.

In the previous section, for each contributing path P (that avoids ≥ 1 failure), we saw that it necessarily diverges from P_0. Since we have multiple paths in $\mathscr{P}_0$, we define some new notations (see Figure 5).

▶ **Definition 13** (Modified P_0 and $D_0(P)$).

1. For any path P (or its corresponding P_1), we define $\tilde{P}_0(P)$ (or $\tilde{P}_0(P_1)$) to be the last path from $\mathscr{P}_0$ which intersects with P (or P_1), say at vertex y, such that at least one of e_1 or e_2 is present in $\tilde{P}_0(P)[y, v]$ (or $e_1 \in \tilde{P}_0(P_1)[y, v]$).
2. For any path P (or P_1), we define $\tilde{D}_0(P) = P[v^*, v]$ (or $\tilde{D}_0(P_1) = P_1[v_0^*, v]$), where v^* is the last vertex of P (or P_1) on $P \cap \tilde{P}_0(P)$ (or $P_1 \cap \tilde{P}_0(P_1)$).

Note that in the single source case, both P and P_1 diverge from the same path P_0. However, in multiple source that path $\tilde{P}_0(P)$ and $\tilde{P}_0(P_1)$ may differ. This is one of the

major changes from the single source case. In fact, the reader will see that all our lemmas in Section 4 extend here with P_0 changed to $\tilde{P}_0(P)$ and $\mathrm{D}_0(P)$ changed to $\tilde{D}_0(P)$. However, for completeness we have re-proven all lemmas.

5.1 Properties of Contributing paths

We now describe important properties of paths in $\mathscr{P}_0$ and contributing paths as follows (see full paper [11] for proofs).

▶ **Lemma 14.** *The set of paths $\mathscr{P}_0$ is converging.*

The number of contributing paths avoiding failures in $\mathscr{P}_0$ can easily bounded to $O(\sqrt{\sigma n})$ for each v (see full paper [11] for details). Excluding these paths, every contributing path satisfies the following properties.

▶ **Lemma 15.** *Excluding $O(\sqrt{\sigma n})$ paths, for any contributing path P from s to v avoiding $\{e_1, e_2\}$, the following properties holds true*
$\mathbb{P}_1$ **:** *$e_1 \in P_0$ and $e_2 \in \mathrm{D}_0(P_1)$.*
$\mathbb{P}_2$ **:** *$\tilde{D}_0(P)$ does not intersect with any path in $\mathscr{P}_0$. Also, if $\tilde{D}_0(P)$ diverges from P_1 it does not intersect it again.*

5.2 Space Analysis

As described earlier, in order to bound the size of dual FT-MBFS subgraph to $O(\sigma^{1/3}n^{5/3})$, it suffices to bound the number of *contributing* paths from $s \in S$ to each vertex $v \in V$ avoiding two edge failures to $O(\sigma^{1/3}n^{2/3})$. Further, using $\mathbb{P}_1$ we are only concerned with a contributing path P if $e_1 \in P_0$ and $e_2 \in \mathrm{D}_0(P_1)$. For the sake of highlighting similarity with single source case, we shall use $n_\sigma = n/\sigma$ throughout the section.

We first divide the paths in $\mathscr{P}_0$ into two parts as follows. For each $s \in S$, let $P_0(s, v)$ be the shortest path from s to v. Let v_{ls} be the vertex such that $|P_0(s, v)[v_{ls}, v]| = n_\sigma^{1/3}$. We define $\mathcal{P}_{low} = \{P_0(s, v)[v_{ls}, v] \mid s \in S\}$ and $\mathcal{P}_{high} = \{P_0(s, v)[s, v_{ls}] \mid s \in S\}$. This definition naturally extends the P_{low} and P_{high} defined in the single source case.

With this modified $\mathcal{P}_{low}$ and $\mathcal{P}_{high}$, we use the same definition of *standard paths* and hence $\mathscr{P}_a$ and $\mathscr{P}_{1a}$. However, the distinction of *long standard paths* ($\mathscr{P}_b$) from *short standard paths* ($\mathscr{P}_c$) would now be done by using $\tilde{D}_0(P_1)$ instead of $D_0(P_1)$. Hence, the *long standard paths* would be the standard paths with $|\tilde{D}_0(P_1)| \geq n_\sigma^{2/3}$. Finally, the definition of $LastPath(P)$ and $LastLeg(P)$ does not change, except in case $LastPath(P) = \phi$, we use $LastPath(P) = \tilde{P}_0(P)$ instead of $LastPath(P) = P_0$ (recall Definition 9). Moreover, the properties of $LastLeg(P)$ also remain same except for Lemma 102 which is modified as follows.

▶ Lemma 10. For every set $\mathscr{P}_x$ (for $x = a, b$ or c), we have the following.
b^***.** Number of paths $P \in \mathscr{P}_x$ with $|LastLeg(P)| > n_\sigma^{\frac{1}{3}}$ or $LastPath(P) = \tilde{P}_0(P)$, is $O(\sigma^{\frac{1}{3}} n^{\frac{2}{3}})$.

Now, using the properties described in Lemma 10 (see full paper [11] for proof), we can analyze the number of *non-standard paths* ($\mathscr{P}_a$) using the same *counting arguments* as in case of single source, bounding the number of such paths to $O(\sigma^{1/3}n^{2/3})$ (see full paper [11] for details). Hence, we only focus on analyzing the *standard paths* ($\mathscr{P}_b$ and $\mathscr{P}_c$) as follows.

Properties of standard paths ($\mathscr{P}_b$ and $\mathscr{P}_c$)

Recall the properties of *standard paths* described in Lemma 11. For multiple sources, Lemma 111 does not hold, because for two paths P and P', their corresponding paths $\mathrm{D}_0(P_1)$

and $\mathrm{D}_0(P_1')$ can diverge after intersection, if they start from different sources say s_1, s_2 (for $s_1 = s_2$, Lemma 111 applies). For example (see Figure 5), P_1 avoids e_1 on $P_0[s_1, v]$. Also, $\mathrm{D}_0(P_1)$ passes through $P_0[s_2, v]$. Let P_1' be a path avoiding e_1' on $P_0[s_2, v] \cap P_1$, such that $\mathrm{D}_0(P_1')$ intersect $\mathrm{D}_0(P_1)$ before $\mathrm{D}_0(P_1)$ enters $P_0[s_2, v]$. Hence, $\mathrm{D}_0(P_1')$ has to diverge from $\mathrm{D}_0(P_1)$ as $\mathrm{D}_0(P_1)$ passes through e_1' after the intersection.

This is the primary reason for defining *modified detour* $\tilde{D}_0(P_1)$, for which a lemma equivalent to Lemma 111 holds. Thus, the analysis of *standard paths* for multiple sources, uses $\tilde{D}_0(P_1)$ instead of $D_0(P_1)$ satisfying the following properties.

▶ **Lemma 16.** *For the set of contributing* standard *paths, we have the following properties.*

(a) *The set of paths* $\{\tilde{D}_0(P_1) | P_1 \in \mathscr{P}_{1b} \cup \mathscr{P}_{1c}\}$*, is converging.*

(b) *The number of paths* $P \in \mathscr{P}_b \cup \mathscr{P}_c$*, which* $Source(LastLeg(P)) \notin \tilde{D}_0(P_1')$ *for some* $P_1' \in \mathscr{P}_{1b} \cup \mathscr{P}_{1c}$ *are* $O(\sigma^{1/3} n^{2/3})$*.*

Using Lemma 162, we only have to bound the number of *standard paths* whose $LastLeg(P)$ originates from some $\tilde{D}_0(P_1')$. Using Lemma 10$b^*$and by definition of long standard paths $\mathscr{P}_b$, Theorem 12 is applicable for the set $\tilde{D}_0(P_1)$ for $P_1 \in \mathscr{P}_{1b}$ and $\alpha = n_\sigma^{1/3}$, bounding number of such paths in $\mathscr{P}_b$ to $O(\sigma^{1/3} n^{2/3})$. This leaves only the number of *short standard paths* that originate from some $\tilde{D}_0(P_1')$ described in the following section.

5.2.1 Analyzing short standard paths $\mathscr{P}_c$

Again, we only analyze the paths in $\mathscr{P}_c$ for undirected graphs here (see full paper [11] for directed graphs). Using Lemma 10$b^*$and Lemma 162, we know that the number of $P \in \mathscr{P}_c$ with $|LastLeg(P)| > n_\sigma^{1/3}$ or $LastPath(P) = \tilde{P}_0(P)$ or $Source(LastLeg(P)) \notin \tilde{D}_0(P_1')$ (for some $P_1' \in \mathscr{P}_{1c}$) is $O(\sigma^{1/3} n^{2/3})$. We thus focus on the case when $|LastLeg(P)| \leq n_\sigma^{1/3}$ and $LastPath(P) \in \mathscr{P}_{1c}$ with $Source(LastLeg(P)) \in \tilde{D}_0(LastPath(P))$. Any such path can be divided into three parts (not necessarily non-empty) including (a) $P[s, x] = P_1[s, x]$, where $x = Source(\tilde{D}_0(P_1))$, (b) $P[x, y]$ where $y = Source(LastLeg(P))$ and (c) $P[y, v] = LastLeg(P)$.

We find alternate paths for $P[s, x]$ and $P[x, y]$, which will help us in bounding their respective lengths (see Figure 6). By definition $\tilde{P}_0(P_1)$ intersects with P_0 and passes through e_1. Further, using Lemma 14 we know that P_0 and $\tilde{P}_0(P_1)$ will merge after the intersection at some point, say x', where $e_1 \in \tilde{P}_0(P_1)[x', v] = P_0[x', v]$. Hence, we have an alternate path for $P_1[s, x]$ avoiding e_1 and e_2 (since $e_2 \notin \mathscr{P}_0$ by $\mathbb{P}_2$) formed by $P_0[s, x'] \cup \tilde{P}_0(P_1)[x', x]$. Now, bounding $P[x, y]$ is exactly same as in the case of single source, using $\tilde{D}_0(P_1)$ instead of $\mathrm{D}_0(P_1)$, bounding $P[x, y]$ to $2n_\sigma^{2/3}$ as shown in Figure 6 (see full paper [11] for an exhaustive proof). Since $P[s, v]$ is the shortest path avoiding $\mathrm{F}(P)$ we have

$$
\begin{aligned}
|P| &= |P_0[s, x]| + |P_1[x, y]| + |P[y, v]| && \text{(by definition of } x \text{ and } y\text{)} \\
&\leq (|P_0[s, x']| + |\tilde{P}_0(P_1)[x', x]|) + 2n_\sigma^{2/3} + n_\sigma^{1/3} && \text{(Similar to dual FT-BFS)} \\
&\leq |P_0[s, v]| + |\tilde{P}_0(P_1)[x, v]| + 2n_\sigma^{2/3} + n_\sigma^{1/3} \\
&\leq |P_0[s, v]| + |\tilde{D}_0(P_1)[x, v]| + 2n_\sigma^{2/3} + n_\sigma^{1/3} \\
&\qquad (\tilde{D}_0(P_1) \text{ is a detour from } \tilde{P}_0(P_1), \text{ hence } |\tilde{D}_0(P_1)[x, v]| > |\tilde{P}_0(P_1)[x, v]|) \\
&\leq |P_0[s, v]| + n_\sigma^{2/3} + 2n_\sigma^{2/3} + n_\sigma^{1/3} && \text{(by definition of } \mathscr{P}_c\text{)}
\end{aligned}
$$

Now, for any $s \in S$, let $\mathscr{P}_c(s)$ be the set of all contributing paths in $\mathscr{P}_c$ that start from s. Using Lemma 112 (that holds for $P \in \mathscr{P}_c(s)$), we know that for any $P, P' \in \mathscr{P}_c(s)$ we have $|P| \neq |P'|$. We thus arrange the paths in $\mathscr{P}_c(s)$ (except the ones in Lemma 10$b^*$and

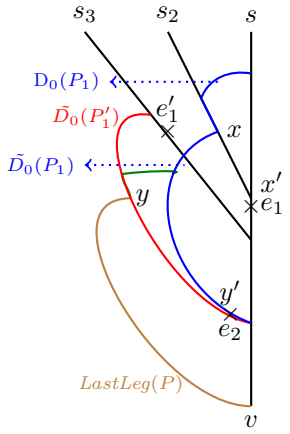


Figure 6 Let $\tilde{P}_0(P_1) = P_0(s_2, v)$, $LastPath(P) = P_1'$ and $\tilde{P}_0(P_1') = P_0(s_3, v)$. Then the path $P_0[s, x'] \cup \tilde{P}_0(P_1)[x', x] \cup \tilde{D}_0(P_1)[x, y'] \cup \tilde{D}_0(P_1')[y', y] \cup LastLeg(P)$ is a valid path avoiding $\{e_1, e_2\}$.

Lemma 162) in the increasing order of sizes, where i^{th} such path has the length $\geq |P_0(s, v)| + i$ (as all paths at least as long as $P_0(s, v)$). Since for any such $P \in \mathscr{P}_c(s)$ we also have $|P| \leq |P_0[s, v]| + 4n^{2/3}$ (described above), clearly the number of paths in $\mathscr{P}_c(s)$ are $O(n_\sigma^{2/3})$ (for i upto $4n^{2/3}$). Hence, overall the number of paths in $\mathscr{P}_c$ considering all sources $s \in S$ are $O(\sigma * n_\sigma^{2/3}) = O(\sigma^{1/3} n^{2/3})$.

This completes the proof of Theorem 4.

6 Conclusion

In this paper, we simplified the analysis in [15] for dual FT-BFS problem and extended it to dual FT-MBFS problem. Unfortunately, extending our result to k FT-MBFS (or even k FT-BFS) problem requires a lot of case analysis. Ideally, one would wish to design a simple data structure to handle multiple failures using some new insight with little or no case analysis. A natural step would be to completely understand these simple cases and derive significant inferences from them to develop new techniques. The simplicity of FT-BFS structure [16] enables a clear understanding of the basic technique used for its construction and analysis. Our work aims to be a significant step to achieve the same for dual FT-BFS by simplifying the result of [15] and generalizing it similar to [16].

References

1 Surender Baswana, Shreejit Ray Chaudhury, Keerti Choudhary, and Shahbaz Khan. Dynamic DFS in Undirected Graphs: breaking the O(m) barrier. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 730–739, 2016.

2 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant reachability for directed graphs. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, pages 528–543, 2015.

3 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant subgraph for single source reachability: generic and optimal. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 509–518, 2016.

4 Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 167–178, 2015.

5 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Multiple-edge-fault-tolerant approximate shortest-path trees. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 18:1–18:14, 2016.

6 Gilad Braunschvig, Shiri Chechik, David Peleg, and Adam Sealfon. Fault tolerant additive and (μ, α)-spanners. *Theor. Comput. Sci.*, 580:94–100, 2015.

7 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault Tolerant Spanners for General Graphs. *SIAM J. Comput.*, 39(7):3403–3423, 2010. doi:10.1137/090758039.

8 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.

9 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 169–178, 2011. doi:10.1145/1993806.1993830.

10 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 506–515, 2009.

11 Manoj Gupta and Shahbaz Khan. Multiple Source Dual Fault Tolerant BFS Trees. *CoRR*, abs/1704.06907, 2017.

12 Neelesh Khanna and Surender Baswana. Approximate shortest paths avoiding a failed vertex: Optimal size data structures for unweighted graphs. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, pages 513–524, 2010.

13 Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, 1979.

14 Merav Parter. Vertex Fault Tolerant Additive Spanners. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, pages 167–181, 2014.

15 Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 481–490, 2015.

16 Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 779–790, 2013.

17 Merav Parter and David Peleg. Fault tolerant approximate BFS structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1073–1092, 2014.

Near-Optimal Induced Universal Graphs for Bounded Degree Graphs

Mikkel Abrahamsen[*1], Stephen Alstrup[†2], Jacob Holm[3], Mathias Bæk Tejs Knudsen[*‡4], and Morten Stöckel[§5]

1 **University of Copenhagen, Copenhagen, Denmark**
`miab@di.ku.dk`
2 **University of Copenhagen, Copenhagen, Denmark**
`s.alstrup@di.ku.dk`
3 **University of Copenhagen, Copenhagen, Denmark**
`jaho@di.ku.dk`
4 **University of Copenhagen, Copenhagen, Denmark**
`mathias@tejs.dk`
5 **University of Copenhagen, Copenhagen, Denmark**
`morten.stockel@gmail.com`

Abstract

A graph U is an induced universal graph for a family $\mathcal{F}$ of graphs if every graph in $\mathcal{F}$ is a vertex-induced subgraph of U.

We give upper and lower bounds for the size of induced universal graphs for the family of graphs with n vertices of maximum degree D. Our new bounds improve several previous results except for the special cases where D is either near-constant or almost $n/2$. For constant even D Butler [Graphs and Combinatorics 2009] has shown $O\big(n^{D/2}\big)$ and recently Alon and Nenadov [SODA 2017] showed the same bound for constant odd D. For constant D Butler also gave a matching lower bound. For generals graphs, which corresponds to $D = n$, Alon [Geometric and Functional Analysis, to appear] proved the existence of an induced universal graph with $(1 + o(1)) \cdot 2^{(n-1)/2}$ vertices, leading to a smaller constant than in the previously best known bound of $16 \cdot 2^{n/2}$ by Alstrup, Kaplan, Thorup, and Zwick [STOC 2015].

In this paper we give the following lower and upper bound of

$$\binom{\lfloor n/2 \rfloor}{\lfloor D/2 \rfloor} \cdot n^{-O(1)} \text{ and } \binom{\lfloor n/2 \rfloor}{\lfloor D/2 \rfloor} \cdot 2^{O\left(\sqrt{D \log D} \cdot \log(n/D)\right)},$$

respectively, where the upper bound is the main contribution. The proof that it is an induced universal graph relies on a randomized argument. We also give a deterministic upper bound of $O\Big(\frac{n^k}{(k-1)!}\Big)$. These upper bounds are the best known when $D \leq n/2 - \tilde{\Omega}(n^{3/4})$ and either D is even and $D = \omega(1)$ or D is odd and $D = \omega\left(\frac{\log n}{\log \log n}\right)$. In this range we improve asymptotically on the previous best known results by Butler [Graphs and Combinatorics 2009], Esperet, Arnaud and Ochem [IPL 2008], Adjiashvili and Rotbart [ICALP 2014], Alon and Nenadov [SODA 2017], and Alon [Geometric and Functional Analysis, to appear].

1998 ACM Subject Classification E.1 Data Structures, 2.2 Graph Theory

[*] Research partly supported by Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research under the Sapere Aude research career programme.
[†] Research partly supported by the FNU project AlgoDisc, Villum Fonden, and the innovationsfonden project DABAI.
[‡] Research partly supported by the FNU project AlgoDisc.
[§] Research partly supported by Villum Fonden and the innovationsfonden project DABAI.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 128; pp. 128:1–128:14

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

Keywords and phrases Adjacency labeling schemes, Bounded degree graphs, Induced universal graphs, Distributed computing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.128

1 Introduction

A graph $G = (V, E)$ is said to be an *induced universal graph* for a family $\mathcal{F}$ of graphs if it contains each graph in $\mathcal{F}$ as a vertex-induced subgraph. A graph $H = (V', E')$ is contained in G as a *vertex-induced subgraph* if $V' \subseteq V$ and $E' = \{vw \mid v, w \in V' \wedge vw \in E\}$. Induced universal graphs have been studied since the 1960s [45, 49], and bounds on the sizes of induced universal graphs have been given for many families of graphs, including general, bipartite [11], and bounded arboricity graphs [10]. In Table 2 in Section 2.3 we give an overview of the state of the art for various graph families along with the results in this paper.

1.1 Adjacency labeling schemes and induced universal graphs

An *adjacency labeling scheme* for a given family $\mathcal{F}$ of graphs assigns *labels* to the vertices of each graph in $\mathcal{F}$ such that a *decoder* given the labels of two vertices from a graph, and no other information, can determine whether or not the vertices are adjacent in the graph. The labels are assumed to be bit strings, and the goal is to minimize the maximum label size. A b-bit labeling scheme uses at most b bits per label. Information theoretical studies of adjacency labeling schemes go back to the 1960s [16, 17], and efficient labeling schemes were introduced in [35, 47]. The first labeling schemes for bounded degree graphs were given in [17]. Let $g_v(\mathcal{F})$ be the smallest number of vertices in any induced universal graph for a family of graphs $\mathcal{F}$. In the families of graphs we study in this paper, a graph always has n vertices, unless explicitly stated otherwise.

A labelling scheme for $\mathcal{F}$ is said to have *unique labels* if no two vertices in the same graph from $\mathcal{F}$ are given the same label. We have the following connection between induced universal graph sizes and label sizes.

▶ **Theorem 1** ([35]). *A family $\mathcal{F}$ of graphs has a b-bit adjacency labeling scheme with unique labels iff $g_v(\mathcal{F}) \leq 2^b$.*

1.2 Our results

The contribution of this paper are stronger bounds on the size of induced universal graphs for bounded degree graphs. Our new bounds are the best known for a significant part of the parameter space, specifically we improve on previous results unless D is either near-constant or almost $n/2$. The best known results for the entire parameter range of induced universal graphs for bounded degree D graphs are shown in Table 1. When the bounded degree D is constant then Butler [18] along with Alon and Nenadov [8] gave optimal bounds. When D is even and of size $\omega(1)$ and when D is odd and of size $\omega(\log n / \log\log n)$ our first new upper bound is the best known as long as $D = O((\log n)\log\log n)$. Let $\mathcal{G}_D$ be the family of graphs with n vertices and maximum degree D. We show the following.

▶ **Theorem 2.** *For the family $\mathcal{G}_D$ of graphs with bounded degree D on n nodes*

$$g_v(\mathcal{G}_D) \leq 8 \cdot \frac{n^k}{(k-1)!}, \quad \text{where } k = \lceil D/2 \rceil .$$

Table 1 The state-of-the-art landscape for induced universal graph sizes. The first column denotes in which range the corresponding bound is the best known.

Range of D	Bound	Reference
D even and $D = O(1)$	$O(n)^{D/2}$	[18]
D odd and $D \in \left[3, O\left(\frac{\log n}{\log\log n}\right)\right]$	$O(D)^D\, n^{D/2}$	[8]
D even and $D \in \left[\omega(1), O\left((\log n)^2 \log\log n\right)\right]$	$O\left(\frac{n^{D/2}}{(D/2-1)!}\right)$	Theorem 2
D odd and $D \in \left[\omega\left(\frac{\log n}{\log\log n}\right), O\left((\log n)^2 \log\log n\right)\right]$	$O\left(\frac{n^{(D+1)/2}}{((D-1)/2)!}\right)$	Theorem 2
$D \in \left[\omega\left((\log n)^2 \log\log n\right), \frac{n}{2} - \Omega\left(n^{3/4}\log^{3/4} n\right)\right]$	$\binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor} 2^{O\left(\sqrt{D\log D}\log(n/D)\right)}$	Theorem 3
$D \geq \frac{n}{2} - O\left(n^{3/4}\log^{3/4} n\right)$	$(1+o(1))2^{(n-1)/2}$	[5]

Our second new upper bound is the smallest induced universal graph for the interval starting where Theorem 2 ends and as long as $D \leq \frac{n}{2} - O\left(n^{3/4}\log^{3/4} n\right)$. The previous best upper bound for such large D was $\binom{n}{\lceil D/2\rceil} n^{O(1)}$ due to Adjiashvili and Rotbart [3]. The bound presented in Theorem 3 is a randomized construction, which works for any D, and which improves asymptotically on Adjiashvili and Rotbart [3] for $D = \omega(1)$. We show the following.

▶ **Theorem 3.** *For the family $\mathcal{G}_D$ of graphs with bounded degree D on $n \geq 2D$ nodes*

$$g_v(\mathcal{G}_D) \leq \binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor} \cdot 2^{O\left(\sqrt{D\log D}\cdot\log(n/D)\right)}.$$

We note that our bound together with the lower bound from Corollary 8 shows that for $D = \omega(1)$, $g_v(\mathcal{G}_D) = \binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor}^{1\pm o(1)}$. In contrast when $D \leq n/2(1-\Omega(1))$ and $D = \Omega(n)$ the bound $\binom{n}{\lceil D/2\rceil} n^{O(1)}$ due to Adjiashvili and Rotbart [3] is $\binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor}^{1+\Omega(1)}$, so we give the first near-optimal induced universal graph when D is superconstant.

From a labeling scheme perspective, the combination of Theorems 2 and 3 shows the existence of an adjacency labeling scheme for $\mathcal{G}_D$ of size

$$\log\binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor} + O\left(\min\left\{D + \log n, \sqrt{D\log D}\log(n/D)\right\}\right).$$

This new labeling scheme improves upon previous in the same ranges as the improvements for the induced universal graphs as shown in Table 1.

In Corollary 8 we show that the any adjacency labeling scheme for $\mathcal{G}_D$ must have labels of size at least $\log\binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor} - O(\log n)$. Our new lower bounds differ from our upper bounds by $O\left(\min\left\{D + \log n, \sqrt{D\log D}\log(n/D)\right\}\right)$, which is at most $O(\sqrt{n\log n})$.

2 Related results

2.1 Maximum degree D

Let $k = \lceil D/2\rceil$. To give an upper bound for any value of D Butler [18] showed the following corollary, which follows from the classic decomposition theorem by Petersen (see [41]):

▶ **Corollary 4** ([18]). *Let $G \in \mathcal{G}_D$ be a graph on n vertices with maximum degree D. Then G can be decomposed into k edge disjoint subgraphs where the maximum degree of each subgraph is at most 2.*

To achieve an upper bound for $g_v(\mathcal{G}_D)$ this can be combined with:

▶ **Theorem 5** ([20]). *Let $\mathcal{F}$ and $\mathcal{Q}$ be two families of graphs and let G be an induced universal graph for $\mathcal{F}$. Suppose that every graph in the family $\mathcal{Q}$ can be edge-partitioned into ℓ parts, each of which forms a graph in $\mathcal{F}$. Then $g_v(\mathcal{Q}) \leq |V[G]|^\ell$.*

Using Theorem 5, Butler [18] concluded that $g_v(\mathcal{G}_D) \leq (6.5n)^k$. Similarly Esperet *et al.* [29] achieved $g_v(\mathcal{G}_D) \leq (2.5n + O(1))^k$, and most recently it was shown by Abrahamsen *et al.* [2] that $g_v(\mathcal{G}_D) \leq (2n-1)^k < 2^k n^k$ due to an induced universal graph for $\mathcal{G}_2$ of size $2n-1$.

For constant maximum degree D, Butler [18] also showed $g_v(\mathcal{G}_D) = \Omega(n^{D/2})$, hence when D is even and constant, $g_v(\mathcal{G}_D) = \Theta(n^{D/2})$ is the right answer up to constant factors due to the above bounds.

2.2 Constant odd degree

A *universal* graph for a family of graphs $\mathcal{F}$ is a graph that contains each graph from $\mathcal{F}$ as a subgraph (not necessarily vertex induced). It is a natural question how to construct universal graphs with as few edges as possible.

A graph has *arboricity* k if the edges of the graph can be partitioned into at most k forests. Graphs with maximum degree D have arboricity bounded by $\lfloor \frac{D}{2} \rfloor + 1$ [19, 40].

When D is odd and constant, some improvement has been achieved on the above bounds on $g_v(\mathcal{G}_D)$ by arguments involving universal graphs and graphs with bounded arboricity [7, 29]. Let $\mathcal{A}_k$ denote a family of graphs with arboricity at most k.

▶ **Theorem 6** ([20]). *Let G be a universal graph for $\mathcal{A}_k$ and d_i the degree of vertex i in G. Then $g_v(\mathcal{A}_k) \leq \sum_i (d_i + 1)^k$.*

Alon and Capalbo [6] described a universal graph with n vertices of maximum degree $c(D)n^{1-2/D}\log^{4/D} n$ for the family $\mathcal{G}_D$, where $D \geq 3$ and $c(D)$ is a constant. Using this bound in Theorem 6, Esperet *et al.* [29] noted that for odd D (and hence arboricity $k = \lceil \frac{D}{2} \rceil$), we get $g_v(\mathcal{G}_D) \leq c_1(D)n^{k-\frac{1}{D}}\log^{2+\frac{2}{D}} n$, for a constant $c_1(D)$.[1] Using the slightly better universal graphs from [7] the maximum degree is reduced to $c(D)n^{1-2/D}$ [4], giving $g_v(\mathcal{G}_D) \leq c_2(D)n^{k-\frac{1}{D}}$, for a constant $c_2(D)$. Note that applying Theorem 6 along with universal graph [7] as above, then for even values of D this would give $g_v(\mathcal{G}_D) \leq c_3(D)n^{\frac{D}{2}+1-\frac{2}{D}}$, for a constant $c_3(D)$. Recently, Alon and Nenadov [8] showed an upper bound $g_v(\mathcal{G}_D) = O(n^{D/2})$, coinciding with Butler's lower bound up to constant factors for any constant D.

2.3 Other graph families

For the family of general, undirected graphs on n vertices, Alstrup *et al.* [11] gave an induced universal graph with $O(2^{n/2})$ vertices, which matches a lower bound by Moon [45]. More recently Alon [5] showed a construction that is tight up to an additive lower order term. We note that whereas the construction of [11] is presented as a labeling scheme, with efficient encoding and constant decoding time. However, it is not obvious if the the induced universal graph from [5] can be transformed into a labeling scheme without requiring that either the encoder or decoder use exponential space or time.

[1] In [29] a typo states that the maximum degree for the universal graph in [6] is $c(D)n^{2-2/D}\log^{4/D} n$. The theorem in [6] only states the total number of edges being $c(D)n^{2-2/D}\log^{4/D} n$, however the maximum degree is $c(D)n^{1-2/D}\log^{4/D} n$ [4].

Table 2 Induced-universal graphs for various families of graphs. "P" is results in this paper. For the max degree results $k = \lceil D/2 \rceil$. In the result for families of graphs with an excluded minor, the $O(1)$ term in the exponent depends on the fixed minor excluded.
* The upper bounds from [11] are labeling schemes with efficient encoding and constant decoding time, but the upper bounds are larger by a constant factor. It is not obvious if the induced universal graph from [5] can be transformed into a labeling scheme without requiring that either the encoder or decoder use exponential space or time.

Graph family	Lower bound	Upper bound	Lower/Upper
General *	$2^{\frac{n-1}{2}}$	$(1+o(1)) \cdot 2^{\frac{n-1}{2}}$	[45]/[5, 11]
Tournaments *	$2^{\frac{n-1}{2}}$	$(1+o(1)) \cdot 2^{\frac{n-1}{2}}$	[46]/[5, 11]
Bipartite *	$(1-o(1)) \cdot 2^{\frac{n}{4}}$	$(1+o(1)) \cdot 2^{\frac{n}{4}}$	[42]/[5, 11]
A: Max degree D	$\binom{\lfloor n/2 \rfloor}{\lfloor D/2 \rfloor} \cdot n^{-O(1)}$	$O\left(\frac{n^k}{(k-1)!}\right)$	P([39, 43, 44])/P
B: Max degree D	$\binom{\lfloor n/2 \rfloor}{\lfloor D/2 \rfloor} \cdot n^{-O(1)}$	$\binom{\lfloor n/2 \rfloor}{\lfloor D/2 \rfloor} \cdot 2^{O(\sqrt{D \log D} \cdot \log(n/D))}$	P([39, 43, 44])/P
C: Constant max degree D	$\Omega(n^{D/2})$	$O(n^{D/2})$	[18]/[8]
Max degree 2	$11 \lfloor n/6 \rfloor$	$2n-1$	[29]/[2]
Acyclic, max degree 2	$\lfloor 3/2n \rfloor$	$\lfloor 3/2n \rfloor$	[2]/[2]
Excluding a fixed minor	$\Omega(n)$	$n^2(\log n)^{O(1)}$	[33]
Planar	$\Omega(n)$	$n^2(\log n)^{O(1)}$	[33]
Planar, constant degree	$\Omega(n)$	$O(n^2)$	[20]
Outerplanar	$\Omega(n)$	$n(\log n)^{O(1)}$	[33]
Outerplanar, constant degree	$\Omega(n)$	$O(n)$	[20]
Treewidth l	$n2^{\Omega(l)}$	$n(\log \frac{n}{l})^{O(l)}$	[33]
Constant arboricity l	$\Omega(n^l)$	$O(n^l)$	[12]/[10]

It follows from [10, 12] that $g_v(\mathcal{A}_k) = \Theta(n^k)$ for the family $\mathcal{A}_k$ of graphs with constant arboricity k and n vertices. Using universal graphs constructed by Babai *et al.* [13], Bhatt *et al.* [14], and Chung *et al.* [21, 22, 23, 24], Chung [20] obtained the best currently known bounds for e.g. induced universal graphs for planar and outerplanar bounded degree graphs.

Labeling schemes are being widely used and well-studied in the theory community: Chung [20] gave labels of size $\log n + O(\log \log n)$ for adjacency labeling in trees, which was improved to $\log n + O(\log^* n)$ [12] and in [15, 20, 31, 32, 36] to $\log n + \Theta(1)$ for various special cases of trees. Finally it was improved to $\log n + \Theta(1)$ for general trees [10].

Using labeling schemes, it is possible to avoid costly access to large global tables and instead only perform local and distributed computations. Such properties are used in applications such as XML search engines [1], network routing and distributed algorithms [26, 27, 30, 51], dynamic and parallel settings [25, 38], and various other applications [37, 48, 50].

A survey on induced universal graphs and adjacency labeling can be found in [11]. See [34] for a survey on labeling schemes for various queries. We give an overview in Table 2 of dominating existing and new results. In the table, "P" refer to a result in this paper and we define $k = \lceil D/2 \rceil$. The "A" and "B" case below represent two different constructions. The upper bound in "B" is a randomized construction.

3 Preliminaries

Let $[n] = \{0, \ldots, n-1\}$, $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$, $\mathbb{N} = \mathbb{N}_1 = \{1, 2, \ldots\}$, and let $\log n$ refer to $\log_2 n$. For a graph G, let $V[G]$ be the set of vertices and $E[G]$ be the set of edges of G, and let $|G| = |V[G]|$ be the number of vertices. We denote the maximum degree of graph G as $\Delta(G)$. For $i \in \mathbb{N}$, let P_i denote a path with i vertices, and for $i > 2$, let C_i denote a simple cycle with i vertices. We let G^2 denote the square of the unweighted graph G, i.e., there is an edge between two nodes in G^2 if they have at most distance two in G. For a boolean statement B we will denote by $[B]$ the value 1 if B is true and 0 otherwise.

Let G and U be two graphs and let $\lambda\colon V[G] \to V[U]$ be an injective function. If λ has the property that $uv \in E[G]$ if and only if $\lambda(u)\lambda(v) \in E[U]$, we say that λ is an *embedding function* of G into U. G is an *induced subgraph* of U if there exists an embedding function of G into U, and in that case, we say that G is *embedded* in U and that U *embeds* G. Let $\mathcal{F}$ be a family of graphs. U is an *induced universal graph* for $\mathcal{F}$ if G is an induced subgraph of U for each $G \in \mathcal{F}$.

4 General D

In this section we present two upper bounds on $g_v(\mathcal{G}_D)$, the number of nodes in the smallest induced universal graph for graphs on n nodes with bounded degree D. In Theorem 2 we give a deterministic construction of an induced universal graph for $\mathcal{G}_D$ that relies on the fact that P_n^2 is a sparse universal graph for $\mathcal{G}_2$.

In Theorem 3 we give a randomized labeling scheme for $\mathcal{G}_D$. For every graph in the family $\mathcal{G}_D$ we give a randomized assignment of labels to the nodes of the graph and show that the labels are short with non-zero probability, thereby showing that there exist short labels for every graph in $\mathcal{G}_D$. This in turn implies an upper bound on $g_v(\mathcal{G}_D)$. Combining the two results shows the existence of an adjacency labeling scheme for $\mathcal{G}_D$ of size $\log \binom{\lfloor n/2 \rfloor \lfloor}{\lfloor D/2 \rfloor} + O\big(\min\big\{D + \log n, \sqrt{D \log D} \log(n/D)\big\}\big)$.

In Section 4.2 we show to use the results by Liebenau and Wormald [39] to give lower bounds on $g_v(\mathcal{G}_D)$. These lower bounds imply that any adjacency labeling scheme for $\mathcal{G}_D$ must have labels of size at least $\log \binom{\lfloor n/2 \rfloor \lfloor}{\lfloor D/2 \rfloor} - O(\log n)$, which means that the upper bounds are tight up to an additive term of size $O\big(\min\big\{D + \log n, \sqrt{D \log D} \log(n/D)\big\}\big)$, which is at most $O(\sqrt{n \log n})$.

4.1 Upper bounds on $g_v(\mathcal{G}_D)$

We present the proof of our first upper bound stated in Theorem 2.

Proof. For a set S we let $S^{\leq \ell}$ denote the set of all subsets of S of size $\leq \ell$. We note that $\big|S^{\leq \ell}\big| \leq 2\frac{|S|^\ell}{\ell!}$ whenever S is finite.

Fix n, D, let $k = \lceil D/2 \rceil$ and let $H_n = P_n^2$ be the square of the path of length n. Identify the vertices of H_n with $[n]$ in the obvious way. Then two nodes i, j in H_n are adjacent if and only if they are different and $|i - j| \leq 2$. We define the graph G to have vertex set $V[G] = [n] \times [2]^2 \times [n]^{\leq k-1}$. For a node $u = (i, x, y, S)$ in G, we define $id(u) = i$. We also

define $N'(u)$ in the following way:

$$N'(u) = \begin{cases} S & x = y = 0 \\ S \cup \{i+1\} & x = 1, y = 0 \\ S \cup \{i+2\} & x = 0, y = 1 \\ S \cup \{i+1, i+2\} & x = y = 1 \end{cases} .$$

There is an edge between u and v in G if $id(u) \in N'(v)$ or $id(v) \in N'(u)$. We proceed to show that G is an induced universal graph for $\mathcal{G}_D$. Let H be a graph in $\mathcal{G}_D$. We will show that H is an induced subgraph of G. By Corollary 4 we know that we can decompose the edges of H into k edge disjoint subgraphs, $H_0, H_1, \ldots, H_{k-1}$, such that each H_i has vertex set $V[H_i] = V[H]$ and maximum degree at most 2. First we order the nodes of H as $u_0, u_1, \ldots, u_{n-1}$ such that all edges (u_i, u_j) in H_0 satisfy $|i - j| \leq 2$. This is possible since H_0 has maximum degree at most 2, and therefore consists of only paths and cycles. We let x_i (resp. y_i) be 1 if there is an edge from u_i to u_{i+1} (resp. u_{i+2}) in H_0. That is:

$$x_i = [(u_i, u_{i+1}) \in E[H_0]] , \; y_i = [(u_i, u_{i+2}) \in E[H_0]] .$$

We now orient the edges of each of the graphs $H_1, \ldots, H_{k-1}$ such that the out degree of each node is at most 1. This is possible since each of H_i has maximum degree at most 2. We let S_i be the set of all u_i's outgoing neighbours in the graphs $H_1, \ldots, H_{k-1}$, and note that S_i contains at most $k - 1$ elements. We let $\lambda : H \to G$ be defined by $\lambda(i) = (i, x_i, y_i, S_i)$. It is now straightforward to check that λ is an embedding function and therefore that H is an induced subgraph of G. Since H was arbitrarily chosen this shows that G is an induced universal graph of $\mathcal{G}_D$. The number of nodes in G is exactly $4n \cdot \left|[n]^{\leq k-1}\right|$ which yields the desired result. ◀

The intuition behind the randomized bound below is the following. Consider placing all n vertices on a circle in a randomly chosen order and rename the vertices with indices $[n]$ following the order on the circle. Now, a vertex $v \in [n]$ remembers its neighbours in the next half of the circle, i.e., v stores all the adjacent vertices among $\{v + 1, \ldots, v + \lceil n/2 \rceil\}$ (where indices are taken modulu n). If two vertices u, v are adjacent, then clearly either u stores the index of v or conversely, hence an adjacency query can be answered. See Figure 1. A standard application of Chernoff bounds implies that vertex v with high probability stores at most $D/2 + O(\sqrt{D \log n})$ indices. However, this can be tightened by a Lovász Local Lemma argument, since each random variable that denote which indices should be stored depend on at most D^2 other such random variables. This allows us to tigthen the number of stored indices to $D/2 + O(\sqrt{D \log D})$, and it follows that there exists an ordering of the points on the circle where every vertex stores that many neighbours and the theorem follows.
We are ready to show Theorem 3.

Proof. Fix n, D and wlog assume that n is odd. For $D \leq \log n$ the result follows from Theorem 2 so assume that $D \geq \log n$. We assume that n and D are bounded from below by a sufficiently large constant. Let G be a graph in $\mathcal{G}_D$, and wlog assume that $V[G] = [n]$. Let $t_0, t_1, \ldots, t_{n-1} \in [n]$ be chosen independently and uniformly at random, and let $\mathrm{id} : [n] \to [n]$ be a bijection that assigns an identifier to each node of G such that

$$(t_i, i) \prec (t_j, j) \Rightarrow \mathrm{id}(i) < \mathrm{id}(j),$$

for all values of $i, j \in [n]$, where $\prec$ is the lexicographical ordering. We construct the function id by sorting the values t_i, and then choosing id to be a bijection such that

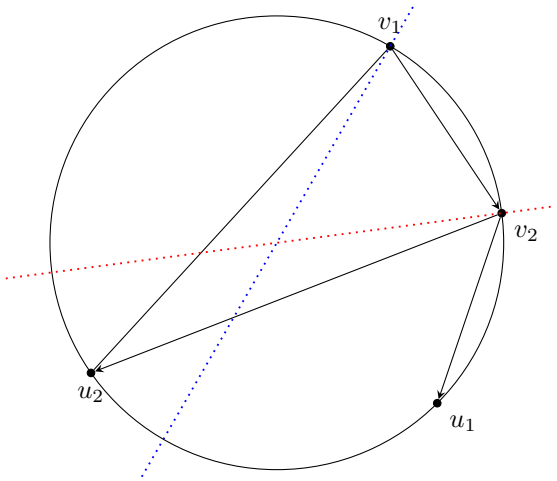


Figure 1 The intuition of the randomized upper bound. Pictured are adjacency relations stored by v_1 and v_2 – an edge denotes adjecency and a directed edge (v, u) denotes v stores its relation to u. Here v_1 stores v_2 but not u_2 as u_2 is on the wrong side of v_1's bisection, and v_2 stores u_1 and u_2, but not v_1 as it is on the wrong side of v_2's bisection.

$(t_{\mathrm{id}(0)}, \mathrm{id}(0)), \ldots, (t_{\mathrm{id}(n-1)}, \mathrm{id}(n-1))$ is a non-decreasing sequence. We note that the values t_i determine id uniquely.

For each $i \in [n]$ we let $S_i \subseteq [n]$ be the set that contains all neighbours j of i for which it holds that:

$$(t_j - t_i) \bmod n \in \left\{0, 1, 2, \ldots, \frac{n-1}{2}\right\}$$

That is, we define S_i by:

$$S_i = \left\{j \in [n] \mid \{i, j\} \in E[G],\ (t_j - t_i) \bmod n \in \left\{0, 1, 2, \ldots, \frac{n-1}{2}\right\}\right\} \tag{1}$$

We say that the values t_i are *good* if the following properties hold for all $i \in [n]$:

$$|S_i| \leq \frac{D}{2} + C\sqrt{D \log D} \tag{2}$$

$$\max_{j \in S_i} \{(\mathrm{id}(j) - \mathrm{id}(i)) \bmod n\} \leq \frac{n}{2} + \max\left\{\frac{n}{D}, C\sqrt{n \log n}\right\} \tag{3}$$

where $C > 0$ is a (sufficiently large) constant to be chosen later. Firstly, we will show that, when choosing the t_i's randomly, they are good with non-zero probability. For $i \in [n]$ let A_i be the event that (2) does not hold for S_i. Let $\mathcal{A} = \{A_i \mid i \in [n]\}$ and for each $A_i \in \mathcal{A}$ let $\Gamma(A_i)$ denote the set of all events A_j where $j \neq i$ has distance at most two to i in G. We note that since G has maximum degree at most D we have that $|\Gamma(A_i)| \leq D^2$. For each $i \in [n]$ the event A_i is independent of all events $\mathcal{A} \setminus (\{A_i\} \cup \Gamma(A_i))$ for the following reason. The event A_i is determined exclusively by the values t_j where $j = i$ or j is a neighbour of i in G. For each A_j such that $A_j \in \mathcal{A} \setminus (\{A_i\} \cup \Gamma(A_i))$ we have that j has distance at least three to i, and A_j is determined by the values $t_{j'}$ where $j' = j$ or j' is a neighbour of j. No such j' can also be a neighbour of i since j has distance three to i and we conclude that A_i is independent of the events $\mathcal{A} \setminus (\{A_i\} \cup \Gamma(A_i))$ for each $i \in [n]$. By a Chernoff bound we have that A_i happens, i.e. (2) is false for S_i, with probability at most $e^{-\Theta(C^2 \log D)}$. Choosing C

sufficiently large this is smaller than $\frac{1}{2}D^{-10}$. Let $x(A_i) = D^{-10}$ for each $i \in [n]$. Then for each $i \in [n]$ we have:

$$x(A_i) \prod_{A_j \in \Gamma(A_i)} (1 - x(A_j)) \geq D^{-10} \left(1 - D^{-10}\right)^{D^2} = D^{-10} e^{-\Theta\left(D^{-10}\right) \cdot D^2}$$
$$> \frac{1}{2} D^{-10} \geq P(A_i)$$

By the Lovász Local Lemma [28, 9] we conclude that the probability that none of $A_i, i \in [n]$ happen is bounded below by

$$\prod_{i \in [n]} (1 - x(A_i)) = \left(1 - D^{-10}\right)^n = e^{-\Theta\left(nD^{-10}\right)}.$$

That is, (2) holds for all S_i with probability at least $e^{-\Theta\left(nD^{-10}\right)}$. For a any value of $i \in [n]$ the probability that (3) is false is at most $\min\left\{e^{-\Theta\left(nD^{-2}\right)}, n^{-\Omega(C^2)}\right\}$ by a standard Chernoff bound. And by a union bound over all choices of $i \in [n]$ (3) holds for all values of i with probability a least $1 - n \min\left\{e^{-\Theta\left(nD^{-2}\right)}, n^{-\Omega(C^2)}\right\}$. Therefore, the probability that the chosen t_is are good is at least

$$e^{-\Theta\left(nD^{-10}\right)} + \left(1 - n \min\left\{e^{-\Theta\left(nD^{-2}\right)}, n^{-\Omega(C^2)}\right\}\right) - 1. \tag{4}$$

We note that (4) is positive if and only if

$$e^{-\Theta\left(nD^{-10}\right)} > n \min\left\{e^{-\Theta\left(nD^{-2}\right)}, n^{-\Omega(C^2)}\right\}, \tag{5}$$

and this can be verified, e.g. by considering the cases $D \leq n^{1/3}$ and $D > n^{1/3}$. That is, the values t_i are good with non-zero probability.

Now fix a good choice of t_i and the corresponding identifier function, id, and the sets S_i. We can now encode the values $\mathrm{id}(i)$ and the set S_i using at most $O(\log n) + \lg \binom{n'}{D'}$ bits where n' and D' are defined by:

$$n' = \left\lfloor \frac{n}{2} + \max\left\{\frac{n}{D}, C\sqrt{n \log n}\right\} \right\rfloor, \quad D' = \left\lfloor \frac{D}{2} + C\sqrt{D \log D} \right\rfloor$$

Let $D'' = \min\left\{\left\lfloor \frac{n'}{2} \right\rfloor, D'\right\}$. Then for any node i we can encode $\mathrm{id}(i)$ and the set $\{(\mathrm{id}(j) - \mathrm{id}(i)) \bmod n \mid j \in S_i\}$ using at most $O(\log n) + \log \binom{n'}{D''}$ bits for the following reason: Firstly, $\mathrm{id}(i)$ can clearly be stored using $O(\log n)$ bits. Secondly, the set of differences $\{(\mathrm{id}(j) - \mathrm{id}(i)) \bmod n \mid j \in S_i\}$ contains at most D' elements which are all contained in $\{1, 2, \ldots, n'\}$, and hence it can be stored using at most $\log\left(n\binom{n'}{D''}\right)$ bits. Given the labels of two nodes i, j we can compute their ids, $\mathrm{id}(i)$ and $\mathrm{id}(j)$, and infer whether $\mathrm{id}(i) \in S_j$ or $\mathrm{id}(j) \in S_i$, i.e. whether i and j are adjacent. Hence we have described a labeling scheme for $\mathcal{G}_D$ using at most $O(\log n) + \log \binom{n'}{D''}$ bits, and therefore

$$g_v(\mathcal{G}_D) \leq \binom{n'}{D''} n^{O(1)}. \tag{6}$$

We first note that:

$$\binom{n'}{D''} \leq \binom{n'}{\lfloor D/2 \rfloor} \cdot \left(\frac{n'}{\lfloor D/2 \rfloor}\right)^{D'' - \lfloor D/2 \rfloor} \leq \binom{n'}{\lfloor D/2 \rfloor} \cdot 2^{O\left(\sqrt{D \log D} \cdot \log(n/D)\right)}. \tag{7}$$

Furthermore we also have:

$$\binom{n'}{\lfloor D/2\rfloor} \leq \binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor} \cdot \left(\frac{n'-\lfloor D/2\rfloor}{\lfloor n/2\rfloor-\lfloor D/2\rfloor}\right)^{\lfloor D/2\rfloor} = \binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor} \cdot \left(1+\frac{n'-\lfloor n/2\rfloor}{\lfloor n/2\rfloor-\lfloor D/2\rfloor}\right)^{\lfloor D/2\rfloor}. \tag{8}$$

We have that $\lfloor n/2\rfloor - \lfloor D/2\rfloor = \Omega(n)$ since $D \leq \frac{n}{2}$ and therefore we get:

$$\left(1+\frac{n'-\lfloor n/2\rfloor}{\lfloor n/2\rfloor-\lfloor D/2\rfloor}\right)^{\lfloor D/2\rfloor} = \left(1+O\left(\frac{n'-\lfloor n/2\rfloor}{n}\right)\right)^{\lfloor D/2\rfloor} \leq e^{O\left(\frac{n'-\lfloor n/2\rfloor}{n}\cdot\lfloor D/2\rfloor\right)}. \tag{9}$$

By the definition of n' we have that

$$\frac{n'-\lfloor n/2\rfloor}{n}\lfloor D/2\rfloor = \max\left\{\frac{1}{D}, C\sqrt{\frac{\log n}{n}}\right\}\cdot\lfloor D/2\rfloor = O\left(\sqrt{D\log D}\right). \tag{10}$$

Combining (8), (9) and (10) we get that

$$\binom{n'}{\lfloor D/2\rfloor} \leq 2^{O\left(\sqrt{D\log D}\right)}\binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor}. \tag{11}$$

Combining (6) with (7) and (11) gives us the desired upper bound on $g_v(\mathcal{G}_D)$

$$g_v(\mathcal{G}_D) \leq \binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor}\cdot 2^{O\left(\sqrt{D\log D}\cdot\log(n/D)\right)}.$$

◀

4.2 Lower bounds on $g_v(\mathcal{G}_D)$

In this section we show how to apply the bounds from [39] on the number of graphs of a given degree sequence. For a graph G with nodes $(u_1, u_2, \ldots, u_n)$ the *degree sequence* of G is $(d_1, d_2, \ldots, d_n)$ where d_i is the degree of u_i. Applying [39, Conjecture 1.1] on a degree sequence $(d, d, \ldots, d)$ we obtain

▶ **Corollary 7** ([39]). *Let n, d be integers such that nd is even and $1 \leq d \leq n-1$. Let $\mu = \frac{d}{n-1}$. The number of d-regular graphs on n nodes is*

$$(1+o(1))\sqrt{2}e^{1/4}\left(\mu^{\mu}(1-\mu)^{1-\mu}\right)^{n(n-1)/2}\binom{n-1}{d}^{n}.$$

We now show that the bound from Corollary 7 implies a lower bound on the size of the induced universal graph for bounded degree graphs:

▶ **Corollary 8.** *For the family $\mathcal{G}_D$ of graphs with bounded degree D on n nodes*

$$g_v(\mathcal{G}_D) = \Omega\left(\sqrt{\frac{1}{\sqrt{D}}\binom{n}{D}}\right). \tag{12}$$

We remark that together with Stirling's approximation, Corollary 8 implies that $g_v(\mathcal{G}_D) \geq \binom{\lfloor n/2\rfloor}{\lfloor D/2\rfloor}n^{-O(1)}$.

Proof. It is clearly enough to prove (12) when $D \leq \left\lfloor \frac{n}{2} \right\rfloor$, since the right hand side is non-increasing for $D \geq \left\lfloor \frac{n}{2} \right\rfloor$. Let $N = 2\left\lfloor \frac{n}{2} \right\rfloor$ be the largest even integer not greater than n. So we assume that $2D \leq N$.

Let X be the number of D-regular graphs on N nodes. By Corollary 7 we have that

$$X = \Theta\left(\left(\mu^\mu (1-\mu)^{1-\mu}\right)^{N(N-1)/2} \binom{N-1}{D}^N \right), \tag{13}$$

where $\mu = \frac{D}{N-1}$. By Stirling's approximation we have that:

$$\begin{aligned}
\binom{N-1}{D} &= \Theta\left(\frac{\sqrt{N-1}\left(\frac{N-1}{e}\right)^{N-1}}{\sqrt{D}\left(\frac{D}{e}\right)^{D}\sqrt{N-1-D}\left(\frac{D}{e}\right)^{D}} \right) \\
&= \Theta\left(\sqrt{\frac{N-1}{D(N-1-D)}} \left(\mu^\mu (1-\mu)^{1-\mu}\right)^{-(N-1)} \right) \\
&= \Theta\left(\sqrt{\frac{1}{D}} \left(\mu^\mu (1-\mu)^{1-\mu}\right)^{-(N-1)} \right).
\end{aligned}$$

Rearranging gives that:

$$\left(\mu^\mu (1-\mu)^{1-\mu}\right)^{(N-1)N/2} = \Theta\left(\sqrt{\frac{1}{D}} \binom{N-1}{D}^{-1} \right)^{N/2}$$

If we insert this into (13) we get that:

$$X = \Theta\left(\sqrt{\frac{1}{D}} \binom{N-1}{D}^{-1} \right)^{N/2} \binom{N-1}{D}^N = \Theta\left(\sqrt{\frac{1}{\sqrt{D}} \binom{N-1}{D}} \right)^N.$$

Since $2D \leq N$ we have that $\binom{N-1}{D} = \Theta\left(\binom{n}{D}\right)$. Clearly X is smaller than $\mathcal{G}_D$, and therefore:

$$|\mathcal{G}_D|^{1/n} = \Omega\left(\sqrt{\frac{1}{\sqrt{D}} \binom{n}{D}} \right) \tag{14}$$

Let G be the induced universal graph for the family $\mathcal{G}_D$. Let $V = [n]$. Any graph H from $\mathcal{G}_D$ on the vertex set V is uniquely defined by the embedding function f of H in G. Since there are no more than $|V[G]|^n$ ways to choose f we get that $|V[G]|^n \geq |\mathcal{G}_D|$. Inserting (14) this shows (12) the following way,

$$g_v(\mathcal{G}_D) = |V[G]| \geq |\mathcal{G}_D|^{1/n} = \Omega\left(\sqrt{\frac{1}{\sqrt{D}} \binom{n}{D}} \right).$$

◀

References

1 S. Abiteboul, H. Kaplan, and T. Milo. Compact labeling schemes for ancestor queries. In *Proc. of the 12th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 547–556, 2001.

2 M. Abrahamsen, S. Alstrup, J. Holm, M. B. T. Knudsen, and M. Stöckel. Near-optimal induced universal graphs for bounded degree graphs. *CoRR*, abs/1607.04911, 2016. URL: `http://arxiv.org/abs/1607.04911`.

3 D. Adjiashvili and N. Rotbart. Labeling schemes for bounded degree graphs. In *41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 375–386, 2014.

4 N. Alon. private communication, 2016.

5 N. Alon. Asymptotically optimal induced universal graphs, 2016. [Online; accessed 5-July-2016]. URL: `http://www.tau.ac.il/~nogaa/PDFS/induniv1.pdf`.

6 N. Alon and M. Capalbo. Sparse universal graphs for bounded-degree graphs. *Random Structures & Algorithms*, 31(2):123–133, 2007.

7 N. Alon and M. Capalbo. Optimal universal graphs with deterministic embedding. In *Proc. of the 19th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 373–378, 2008.

8 N. Alon and R. Nenadov. Optimal induced universal graphs for bounded-degree graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1149–1157, 2017. `doi:10.1137/1.9781611974782.74`.

9 N. Alon and J. H. Spencer. *The probabilistic method*. Wiley Publishing, 2000.

10 S. Alstrup, S. Dahlgaard, and M. B. T. Knudsen. Optimal induced universal graphs and labeling schemes for trees. In *Proc. 56th Annual Symp. on Foundations of Computer Science (FOCS)*, 2015.

11 S. Alstrup, H. Kaplan, M. Thorup, and U. Zwick. Adjacency labeling schemes and induced-universal graphs. In *Proc. of the 47th Annual ACM Symp. on Theory of Computing (STOC)*, pages 625–634, 2015.

12 S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *Proc. 43rd Annual Symp. on Foundations of Computer Science (FOCS)*, pages 53–62, 2002.

13 L. Babai, F. R. K. Chung, P. Erdös R. L. Graham, and J. Spencer. On graphs which contain all sparse graphs. *Ann. discrete Math.*, 12:21–26, 1982.

14 S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg. Universal graphs for bounded-degree trees and planar graphs. *SIAM J. Discrete Math.*, 2(2):145–155, 1989.

15 N. Bonichon, C. Gavoille, and A. Labourel. Short labels by traversal and jumping. In *Structural Information and Communication Complexity*, pages 143–156. Springer, 2006. Include proof for binary trees and caterpillars.

16 M. A. Breuer. Coding the vertexes of a graph. *IEEE Trans. on Information Theory*, IT–12:148–153, 1966.

17 M. A. Breuer and J. Folkman. An unexpected result on coding vertices of a graph. *J. of Mathemathical analysis and applications*, 20:583–600, 1967.

18 S. Butler. Induced-universal graphs for graphs with bounded maximum degree. *Graphs and Combinatorics*, 25(4):461–468, 2009. `doi:10.1007/s00373-009-0860-x`.

19 G. Chartrand, H. V. Kronk, and C. E. Wall. The point-arboricity of a graph. *Israel J. of Mathematics*, 6(2):169–175, 1968. `doi:10.1007/BF02760181`.

20 F. R. K. Chung. Universal graphs and induced-universal graphs. *J. of Graph Theory*, 14(4):443–454, 1990.

21 F. R. K. Chung and R. L. Graham. On graphs which contain all small trees. *J. of combinatorial theory, Series B*, 24(1):14–23, 1978.

22 F. R. K. Chung and R. L. Graham. On universal graphs. *Ann. Acad. Sci.*, 319:136–140, 1979.

23 F. R. K. Chung and R. L. Graham. On universal graphs for spanning trees. *J. London Math. Soc.*, 27:203–211, 1983.

24 F. R. K. Chung, R. L. Graham, and N. Pippenger. On graphs which contain all small trees ii. *Colloquia Mathematica*, pages 213–223, 1976.

25 E. Cohen, H. Kaplan, and T. Milo. Labeling dynamic XML trees. *SIAM J. Comput.*, 39(5):2048–2074, 2010. doi:10.1137/070687633.

26 L. J. Cowen. Compact routing with minimum stretch. *J. of Algorithms*, 38:170–183, 2001. See also SODA'91.

27 T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. *J. of Algorithms*, 46(2):97–114, 2003. doi:10.1016/S0196-6774(03)00002-6.

28 P. Erdos and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10(2):609–627, 1975.

29 L. Esperet, A. Labourel, and P. Ochem. On induced-universal graphs for the class of bounded-degree graphs. *Inf. Process. Lett.*, 108(5):255–260, 2008. doi:10.1016/j.ipl.2008.04.020.

30 P. Fraigniaud and C. Gavoille. Routing in trees. In 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 757–772, 2001.

31 P. Fraigniaud and A. Korman. On randomized representations of graphs using short labels. In *Proc. of the 21st Annual Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 131–137, 2009. doi:10.1145/1583991.1584031.

32 P. Fraigniaud and A. Korman. Compact ancestry labeling schemes for XML trees. In *Proc. of the 21st annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 458–466, 2010.

33 C. Gavoille and A. Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *Algorithms–ESA*, pages 582–593. Springer, 2007.

34 C. Gavoille and D. Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2-3):111–120, 2003. doi:10.1007/s00446-002-0073-5.

35 S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM J. Disc. Math.*, 5(4):596–603, 1992. See also STOC'88.

36 H. Kaplan, T. Milo, and R. Shabo. A comparison of labeling schemes for ancestor queries. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 954–963, 2002.

37 A. Korman. Labeling schemes for vertex connectivity. *ACM Trans. Algorithms*, 6(2):39:1–39:10, 2010. doi:10.1145/1721837.1721855.

38 A. Korman and D. Peleg. Labeling schemes for weighted dynamic trees. *Inf. Comput.*, 205(12):1721–1740, 2007.

39 A. Liebenau and N. Wormald. Asymptotic enumeration of graphs by degree sequence, and the degree sequence of a random graph. *arXiv preprint arXiv:1702.08373*, 2017.

40 L. Lovasz. On decomposition of graphs. *Studia Sci. Math. Hungar*, 1:237–238, 1966.

41 L. Lovász and M.D. Plummer. *Matching Theory*. AMS Chelsea Publishing Series. American Mathematical Soc., 2009.

42 V. V. Lozin and G. Rudolf. Minimal universal bipartite graphs. *Ars Comb.*, 84, 2007.

43 B. D. McKay and N. C. Wormald. Asymptotic enumeration by degree sequence of graphs of high degree. *European Journal of Combinatorics*, 11(6):565–580, 1990.

44 B. D. McKay and N. C. Wormald. Asymptotic enumeration by degree sequence of graphs with degreeso (n 1/2). *Combinatorica*, 11(4):369–382, 1991.

45 J. W. Moon. On minimal n-universal graphs. *Proc. of the Glasgow Mathematical Association*, 7(1):32–33, 1965.

46 J. W. Moon. *Topics on tournaments*. Holt, Rinehart and Winston, 1968.

47 J. H. Müller. *Local structure in graph classes*. PhD thesis, Georgia Institute of Technology, 1988.

48 D. Peleg. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, pages 579–588, 2000.

49 R. Rado. Universal graphs and universal functions. *Acta. Arith.*, 9:331–340, 1964.

50 N. Santoro and R. Khatib. Labeling and implicit routing in networks. *The computer J.*, 28:5–8, 1985.

51 M. Thorup and U. Zwick. Approximate distance oracles. *J. of the ACM*, 52(1):1–24, 2005. See also STOC'01.

Universal Framework for Wireless Scheduling Problems*†

Eyjólfur I. Ásgeirsson[1], Magnús M. Halldórsson[2], and Tigran Tonoyan[3]

1 **School of Science and Engineering, Reykjavik University, Reykjavik, Iceland**
eyjo@ru.is
2 **ICE-TCS, School of Computer Science, Reykjavik University, Reykjavik, Iceland**
mmh@ru.is
3 **ICE-TCS, School of Computer Science, Reykjavik University,Reykjavik, Iceland**
tigran@ru.is

Abstract

An overarching issue in resource management of wireless networks is assessing their *capacity*: *How much communication can be achieved in a network, utilizing all the tools available: power control, scheduling, routing, channel assignment and rate adjustment?* We propose the first framework for approximation algorithms in the physical model that addresses these questions in full, including rate control. The approximations obtained are doubly logarithmic in the link length and rate diversity. Where previous bounds are known, this gives an exponential improvement.

A key contribution is showing that the complex interference relationship of the physical model can be simplified into a novel type of amenable conflict graphs, at a small cost. We also show that the approximation obtained is provably the best possible for any conflict graph formulation.

1998 ACM Subject Classification C.2.1 Network Architecture and Design

Keywords and phrases Wireless, Scheduling, Physical Model, Approximation framework

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.129

1 Introduction

The effective use of wireless networks revolves around utilizing fully all available diversity. This can include power control, scheduling, routing, channel assignment and *transmission rate control* on the links, the latter being an issue of key interest for us. The long-studied topic of *network capacity* deals with how much communication can be achieved in a network when its resources are utilized to the fullest. This can be formalized in different ways, involving a range of problems. The communication ability of packet networks is characterized by the capacity region, i.e. the set of traffic rates that can be supported by any scheduling policy. In order to achieve *low delays and optimal throughput*, the classic result of Tassiulas and Ephremides [28] and followup work in the area (e.g. [25]) point out a core optimization problem that lies at the heart of such questions – the *maximum weight independent set of links (*Mwisl*)* problem: from a given set of communication links with associated weights/utilities, find an *independent (conflict-free, subject to the interference model) subset of maximum total weight.*

* A full version of the paper is available at https://arxiv.org/abs/1705.10104.
† This work was supported by grants 152679-05 and 174484-05 from the Icelandic Research Fund.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 129; pp. 129:1–129:15


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

This reduction applies to very general settings involving single-hop and multi-hop, as well as fixed and controlled transmission rate networks. Moreover, approximating MWISL within any factor implies achieving the corresponding fraction of the capacity region. This makes MWISL a central problem in the area. Unfortunately, solving this problem in its full generality is notoriously hard, since it is well known that MWISL is effectively inapproximable (under standard complexity theory) e.g. in models described by general conflict relations or general graphs. Moreover, in general, even approximating the capacity region in polynomial time within a non-trivial bound, while keeping the delays in reasonable bounds, is hard under standard assumptions [27].

We tackle this question in the *physical model* of communication. Towards this end, we develop a general approximation framework that not only helps us to approximate MWISL, but can also be used for tackling various other scheduling problems, such as TDMA scheduling, joint routing and scheduling and others. The problems handled can additionally involve path or flow selection, multiple channels and radios, and packet scheduling. We obtain *double-logarithmic* (in link and rate diversity) approximation for these problems, exponentially improving the previously known logarithmic approximations, and, importantly, extending them to incorporate *different fixed rates and rate control*. The crucial feature of the approach (which makes it so general) is that it involves transforming the complex physical model into an unweighted/undirected conflict graph and solving the problems simply on these graphs. Perhaps surprisingly, we find that our schema attains the best possible performance of *any* conflict graph representation. Numerical simulations show that the conflict graph framework is a good approximation for the physical model on randomly placed network instances as well. Our approach also finesses the task of selecting optimum power settings by using *oblivious* power assignment, one that depends only on the properties of the link itself and not on other links. The performance bounds are however in comparison with the optimum solution that can use arbitrary power settings.

Technically, our approach generalizes our earlier framework [14]. Our extensions required substantial changes throughout the whole body of arguments. That formulation works only for uniform constant rates, and the generalization requires substantial new ideas. One indicator of the challenges overcome is that we could prove that our doubly-logarithmic approximation is best possible in the presence of different rates, while better approximations are known to hold in the case of uniform rates [14].

We make some undemanding assumptions about the settings. We assume that the networks are *interference-constrained*, in that interference rather than the ambient noise is the determining factor of proper reception. This assumption is common and is particularly natural in settings with rate control, since the impact of noise can always be made negligible by avoiding the highest rates, losing only a small factor in performance. We also assume that nodes are (arbitrarily) located in a doubling metric, which generalizes Euclidean space, allowing the modeling of some of non-geometric effects seen in practice.

Our Results. Our results can be summarized as follows:

- We establish a general framework for tackling wireless scheduling and related problems,
- Our approximations hold for nearly all such problems, including variable rates settings,
- We obtain exponential improvement over previously known approximations,
- The approximations are obtained via simple conflict graphs, as opposed to the complicated physical model, and by using *oblivious* power assignments,
- We establish tight bounds indicating the limitations of our method.

Related work. Gupta and Kumar introduced the physical model of interference/communication with log-path fading in their influential paper [10], and it has remained the default in

analytic studies. Moscibroda and Wattenhofer [26] initiated worst-case analysis of scheduling problems in networks of arbitrary topology, which is also the setting of interest in this paper. There has been significant progress in understanding scheduling problems with fixed uniform rates. NP-completeness results have been given for different variants [8, 21, 24]. Early work on approximation algorithms involve (directly or indirectly) partitioning links into length groups, which results in performance guarantees that are at least logarithmic in Δ, the link length diversity: TDMA scheduling and uniform weights MWISL [8, 5, 11], non-preemptive scheduling [7], joint power control, scheduling and routing [4], and joint power control, routing and throughput scheduling in multiple channels [2], to name a few. Constant-factor approximations are known for uniform weight MWISL (in restricted power regimes [12] and (general) power control [22]). Standard approaches translate the constant-factor approximations for the uniform weight MWISL into $O(\log n)$ approximations for TDMA scheduling and general MWISL. Many problems become easier, including MWISL and TDMA scheduling, in the regime of linear power assignments [6, 33, 13, 29]. Recently, a $O(\log^* \Delta)$-approximation algorithm was given for TDMA scheduling and MWISL [14], by transforming the physical model into a conflict graph. We build on this approach, and extend it into a general framework that covers other problems and incorporates support for rate control.

Very few results are known for problems involving rate control. The constant-factor approximation for MWISL with uniform weights and arbitrary but fixed rates proposed by Kesselheim [23] can be used to obtain $O(\log n)$-approximations for TDMA scheduling and MWISL with rate control, where n is the number of links. Another recent work [9] handles the TDMA scheduling problem (with fixed but different rates), obtaining an approximation independent of the number of links n, but the ratio is polynomial in Δ. There have been numerous algorithms that try to approximate or replace MWISL in the context of packet scheduling. Several examples include Longest-Queue-First Scheduling (LQF) [20], Maximal Scheduling [34], Carrier Sense Multiple Access (CSMA) [19]. The approximations obtained usually depend on some parameter of the conflict graph, such as the *interference degree*. In the case of CSMA (and other similar protocols), it is known that the algorithms are throughput-optimal, but in general they take exponential time to stabilize, or otherwise require constant degree conflict graphs [18]. It is also well known that many scheduling problems such as vertex coloring and MWISL are easy to approximate in bounded *inductive independence* graphs, such as geometric intersection graphs or protocol model. However, fidelity to the cumulative nature of interference and the question of modeling rate control are among the significant issues faced by such graph models.

Paper Organization. The fundamental ideas of our approximation framework are described in Section 2. After introducing the model and definition in Section 3, we derive the core technical part, the approximation of the physical model by the conflict graphs, in Section 4, and the optimality of approximation. The framework is applied to obtain approximations for fixed rate scheduling problems in Section 5 and for problems with rate control in Section 6 (the latter two can be read separately from Section 4). Due to space constraints, several technical proofs are deferred to the full version of this paper.

2 Approximation Method

Before defining the details, let us describe the main idea behind the approximation technique. In essence, we define a notion of approximation of an independence system[1] $\mathcal{I}_{\mathcal{P}} = (L, \mathcal{E}_{\mathcal{P}})$ by

[1] An independence system $\mathcal{I}$ over a set of vertices V is a pair $\mathcal{I} = (V, \mathcal{E})$, where $\mathcal{E} \subseteq 2^V$ is a collection of subsets of vertices that is closed under subsetting: if $S \in \mathcal{E}$ and $S' \subset S$, then $S' \in \mathcal{E}$.

a graph $\mathcal{G} = (L, E)$ over the set L of links. The system $\mathcal{I}_\mathcal{P}$ corresponds to the cumulative interference in the physical model, while $\mathcal{G}$ is a conflict graph describing pairwise conflicts between links. We will refer to independent sets in $\mathcal{I}_\mathcal{P}$ as *feasible sets*, and to independent sets in $\mathcal{G}$ as *independent sets*, to avoid confusion.

The approximation is described by several key properties.

Refinement (Feasibility of Independent Sets). Every independent set S in $\mathcal{G}$ must be feasible, i.e. $S \in \mathcal{E}_\mathcal{P}$. Thus, finding an independent set in $\mathcal{G}$ gives also a feasible set in $\mathcal{I}_\mathcal{P}$.

Tightness (of refinement). There is a small number k such that every feasible set $S \in \mathcal{I}_\mathcal{P}$ is a union of at most k independent sets in $\mathcal{G}$. The smallest such k is called the *tightness* of refinement. This property guarantees that even an optimal (for a problem in question) feasible set can be covered with a few independent sets.

The two properties above establish a tight connection between the two models. That allows us to take nearly *every* scheduling problem in the physical model and reduce it to the corresponding problem in conflict graphs (in a way formalized in Section 5), by paying only an approximation factor depending on the tightness k. However, in order for this scheme to work, it should be easier to solve such problems in $\mathcal{G}$, which leads to the third key property.

Computability. There are efficient (approximation) algorithms for scheduling-related problems such as *vertex coloring* and *maximum weight independent set* in $\mathcal{G}$.

A graph $\mathcal{G}$ satisfying the properties above is said to be a *refinement* of $\mathcal{I}_\mathcal{P}$. The main effort in the following two sections is to define an appropriate conflict graph refinement for the physical model and prove these key properties. We find such a family that approximates the physical model with nearly constant tightness, i.e. double-logarithmic in length and rate diversity and show that this is best possible for any conflict graph, up to constant factors. This approximation allows us to bring to bear the large body of theory of graph algorithms, greatly simplifying both the exposition and the analysis.

3 Model

In scheduling problems, the basic object of consideration is a set L of n communication links, numbered from 1 to n, where each link $i \in L$ represents a single-hop communication request between two wireless nodes located in a metric space – a sender node s_i and receiver node r_i.

We assume the nodes are located in a metric space with distance function d. We denote $d_{ij} = d(s_i, r_j)$ and $l_i = d(s_i, r_i)$. The latter is called the *length* of link i. Let $d(i, j)$ denote the minimum distance between the nodes of links i and j.

The nodes have adjustable transmission power levels. A *power assignment* for the set L is a function $P : L \to \mathbb{R}_+$. For each link i, $P(i)$ defines the power level used by the sender node s_i. In the physical model of communication, when using a power assignment P, a transmission of a link i is successful if and only if

$$SIR(S, i) = \frac{P(i)/l_i^\alpha}{\sum_{j \in S \setminus \{i\}} P(j)/d_{ji}^\alpha} \geq \beta_i, \tag{1}$$

where $\beta_i > 1$ denotes the minimum signal to noise ratio required for link i, $\alpha \in (2, 6)$ is the path loss exponent and S is the set of links transmitting concurrently with link i. Note that we omit the noise term in the formula above, since we focus on interference limited networks. This can be justified by the fact that one can simply slightly decrease the transmission rates

to make the effect of the noise negligible, then restore the rates by paying only constant factors in approximation.

A set S of links is called *P-feasible* if the condition (1) holds for each link $i \in S$ when using power assignment P. We say S is *feasible* if there exists a power assignment P for which S is P-feasible.

Effective Length. Let us denote $\mathfrak{l}_i = \beta_i^{1/\alpha} l_i$ and call it the *effective length* of link i. Let $\Delta(L) = \max_{i,j \in L}\{\mathfrak{l}_i/\mathfrak{l}_j\}$ denote the *(effective) length diversity* of instance L. We call a set S of links *equilength* if for every two links $i, j \in S$, $\mathfrak{l}_i \leq 2\mathfrak{l}_j$, i.e., $\Delta(S) \leq 2$. Note that with the introduction of effective length, the feasibility constraint (1) becomes: $\frac{P(i)}{\mathfrak{l}_i^\alpha} \geq \sum_{j \in S\setminus\{i\}} \frac{P(j)}{d_{ji}^\alpha}$. This looks like the same formula but with uniform rates $\beta_i' = 1$. However, there is an essential difference between the two: the quantities $\mathfrak{l}_i$ are not related to the metric space in the same way as lengths l_i, as $\mathfrak{l}_i$ can be arbitrarily larger than l_i.

Metrics. The *doubling dimension* of a metric space is the infimum of all numbers $\delta > 0$ such that for every ϵ, $0 < \epsilon \leq 1$, every ball of radius $r > 0$ has at most $C\epsilon^{-\delta}$ points of mutual distance at least ϵr where $C \geq 1$ is an absolute constant. For example, the m-dimensional Euclidean space has doubling dimension m [16]. We let m denote the doubling dimension of the space containing the links. We will assume $\alpha > m$, which is the standard assumption $\alpha > 2$ in the Euclidean plane.

4 Conflict Graph Approximation of Physical Model

In this section we present the $O(\log\log\Delta)$-tight refinement of the physical model by conflict graphs. The first part introduces our conflict graph $\mathcal{G}_f$ that generalizes the conflict graph definition of [14] and extends it to general thresholds/rates. The three subsequent parts give the proofs of the three key properties: refinement, tightness and computability. The last part argues the asymptotic optimality of $O(\log\log\Delta)$-tightness for any conflict graph, which contrasts the $O(\log^*\Delta)$ bound known in the uniform thresholds setting.

▶ **Theorem 1.** *There is an $O(\log\log\Delta)$-tight refinement of the physical model by a conflict graph family $\mathcal{G}(L)$.*

Conflict Graphs. We define the conflict graph family as follows.

▶ **Definition 2.** Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a positive non-decreasing function. Links i, j are said to be *f-independent* if $d_{ij}d_{ji} > \mathfrak{l}_i\mathfrak{l}_j f\left(\mathfrak{l}_{max}/\mathfrak{l}_{min}\right)$, where $\mathfrak{l}_{min} = \min\{\mathfrak{l}_i, \mathfrak{l}_j\}, \mathfrak{l}_{max} = \max\{\mathfrak{l}_i, \mathfrak{l}_j\}$, and otherwise *$f$-adjacent*. A set of links is f-independent (f-adjacent) if they are pairwise f-independent (f-adjacent).

The conflict graph $\mathcal{G}_f(L)$ of a set L of links is the graph with vertex set L, where two vertices are adjacent if and only if they are f-adjacent.

This definition extends the conflict graphs introduced in [14], where the independence criterion was $d(i,j)/l_{min} > f(l_{max}/l_{min})$ (l_{max}, l_{min} are the length of the longer and shorter links, resp.). When all threshold values β_i are constant, the latter essentially follows from the definition above by "canceling" l_{max} with the larger value of d_{ij}, d_{ji} (modulo constant factors). In general, however, the effective lengths can be very different from the actual link lengths, and feasibility requires more separation than given by graphs involving distances only. A technical difficulty introduced by the new definition is that we have to keep track of two distances d_{ij} and d_{ji} instead of the single distance $d(i,j)$, but this appears to be necessary.

We will be particularly interested in *sub-linear* functions $f(x) = O(x)$. A function f is *strongly sub-linear* if for each constant $c \geq 1$, there is a constant c' such that $cf(x)/x \leq f(y)/y$ for all $x, y \geq 1$ with $x \geq c'y$. Note that if f is strongly sub-linear then $f(x) = o(x)$. For example, the functions $f(x) = x^\delta$ $(\delta < 1)$ and $f(x) = \log x$ are strongly sub-linear.

Refinement: Feasibility of Independent Sets. Our goal now is to find a function f such that each independent set in $\mathcal{G}_f$ is feasible. It is clear that this can be achieved by letting f grow sufficiently fast. But we should not let it grow too fast, so as to not affect tightness. We also need to indicate which power assignment makes the independent sets in $\mathcal{G}_f$ feasible. Our approach is to preselect a family of *oblivious power assignments*, that are local to each link and do not depend on others, and then find an appropriate function f. Consider the family of power assignments P_τ parameterized by $\tau \in (0, 1)$, where $P_\tau(i) \sim \ell_i^{\tau\alpha}$ for each link i. In order to obtain P_τ-feasibility, we take conflict graphs $\mathcal{G}_f$ with $f(x) = \gamma x^\delta$ for $\delta \in (0, 1)$ and $\gamma \geq 1$. Such graphs are denoted as $\mathcal{G}_\gamma^\delta$. We show that every independent set in $\mathcal{G}_\gamma^\delta$ for appropriate γ and δ is P_τ-feasible for some τ.

▶ **Theorem 3.** *Let $\delta_0 = \frac{\alpha-m+1}{2(\alpha-m)+1}$. If $\delta \in (\delta_0, 1)$ and the constant $\gamma > 1$ is large enough, there is a value $\tau \in (0, 1)$ such that each independent set in $\mathcal{G}_\gamma^\delta$ is P_τ-feasible.*

The proof is an adaptation of the ideas used in the proof of [15, Cor. 6] to our definition of conflict graphs and effective lengths. Given an independent set S in $\mathcal{G}_\gamma^\delta$ and a link i, we bound the interference of S on i by first splitting S into equilength subsets, bounding the contribution of each subset separately, then combining the bounds into one. The core of the proof is a careful application of a common packing argument in doubling metric spaces.

Tightness of Refinement. Now, let us bound the number of f-independent sets that are necessary to cover a feasible set. We show that this number is $O(f^*(\Delta(S)))$ for any feasible set S, where f^* is defined for every strongly sub-linear function, as follows. For each integer $c \geq 1$, the function $f^{(c)}(x)$ is defined inductively by: $f^{(1)}(x) = f(x)$ and $f^{(c)}(x) = f(f^{(c-1)}(x))$. Let $x_0 = \inf\{x \geq 1, f(x) < x\} + 1$; such a point exists for every $f(x) = o(x)$. The function $f^*(x)$, is defined by: $f^*(x) = \arg\min_c\{f^{(c)}(x) \leq x_0\}$ for arguments $x > x_0$, and $f^*(x) = 1$ for the rest. Note that for a function $f(x) = \gamma x^\delta$ with constants $\gamma > 0$ and $\delta \in (0, 1)$, $f^*(\Delta) = \Theta(\log\log\Delta)$, which is the tightness bound we are aiming for.

▶ **Theorem 4.** *Consider a non-decreasing strongly sub-linear function f. Every feasible set S can be split into $O(f^*(\Delta(S)))$ subsets, each independent in $\mathcal{G}_f(S)$.*

Let us fix a function f with properties as in the theorem. We establish the partition in Thm. 4 in two steps. The first step is to show that feasible set S can be partitioned into a constant number of independent sets in $\mathcal{G}_\rho^0(S)$ for any constant ρ, i.e., subsets S' such that for every pair of links $i, j \in S'$, $d_{ij}d_{ji} > \rho\ell_i\ell_j$. Such subsets are called ρ-independent for short. The second step is to show that for an appropriate constant ρ, each ρ-independent set can be partitioned into at most $O(f^*(\Delta))$ of f-independent subsets.

The first step is easy. Each feasible set can be partitioned into at most $2\rho^{\alpha/2}$ subsets, each of them feasible with updated thresholds $\{\rho^{\alpha/2}\beta_i\}$. This is a direct application of Corollary 2 of [3]. Let S' be such a subset and let $i, j \in S'$. The feasibility constraint for i and j implies:

$$P(i)/l_i^\alpha \geq \rho^{\alpha/2}\beta_i P(j)/d_{ji}^\alpha \text{ and } P(j)/l_j^\alpha \geq \rho^{\alpha/2}\beta_j P(i)/d_{ij}^\alpha.$$

By multiplying together the inequalities above, canceling $P(i)$ and $P(j)$ and raising to the power of $1/\alpha$, we obtain: $d_{ij}d_{ji} \geq \rho\ell_i\ell_j$, as required.

The proof of the second step requires the following lemmas, which constitute the most significant technical difference from the proof of the corresponding theorem in [14], as they encapsulate the technicalities of dealing with our definition of conflict graphs: It is not sufficient to bound only one of the distances between links (such as $d(i,j)$ in [14]); we need a bound on the product of two distances.

▶ **Lemma 5.** *Let i,j,k be such that $\ell_i \le \ell_j \le \ell_k$ and i is f-adjacent with both j and k, where f is a non-decreasing sublinear function. Then*

$$d_{jk}d_{kj} < 18\ell_i\ell_k f(\ell_k/\ell_i) + 13\ell_j\ell_k + 2\ell_j\sqrt{\ell_i\ell_k f(\ell_k/\ell_i)} + \ell_k\sqrt{\ell_i\ell_j f(\ell_j/\ell_i)}.$$

▶ **Lemma 6.** *Let i be a link and $\rho > 1$. If E is a ρ-independent set of links where each $j \in E$ is f-adjacent with i and satisfies $\ell_i \le \ell_j \le c\ell_i$ for a constant c, then $|E| = O(1)$.*

Proof of Theorem 4. By the discussion above, it is sufficient to show that each ρ-independent set S, for appropriate constant $\rho > 1$, can be partitioned into a small number of f-independent sets. We choose $\rho = 3c_f + 31$, where c_f is such that $f(x) \le c_f x$ for all $x \ge 1$ (recall that f is sub-linear). Partitioning is done by the following *inductive* coloring procedure: 1. Consider the links in a non-increasing order by effective length, 2. Assign each link the first natural number that has not been assigned to an f-adjacent link yet. Clearly, such a procedure defines a partitioning of S into f-independent subsets.

Fix a link $i \in S$. Let T denote the set of links in $j \in S$ that have greater effective length than i and are f-adjacent with i. In order to complete the proof, it is enough to show that $|T| = O(f^*(\Delta))$, as $|T|$ is an upper bound on the number assigned to link i.

Since $f(x)$ is strongly sub-linear, there exists $x_0 = \inf\{x \ge 1, f(x) < x\} + 1$. Let us split T into two subsets A and B, where A contains the links $j \in T$ such that $\ell_j \le x_0\ell_i$ and $B = T \setminus A$. By Lemma 6, we have that $|A| = O(1)$, so we concentrate on B.

Let j,k be arbitrary links in B such that $\ell_j \le \ell_k$. By applying Lemma 5 and using the definition of c_f, we obtain: $d_{jk}d_{kj} < 18\ell_i\ell_k f(\ell_k/\ell_i) + (3c_f + 13)\ell_j\ell_k$. Recall that j and k are $(\rho = 3c_f + 31)$-independent, so $d_{jk}d_{kj} > (3c_f + 31)\ell_j\ell_k$, which gives us $\ell_j/\ell_i < f(\ell_k/\ell_i)$. Let $1,2,\ldots,t = |B|$ be an arrangement of the links in B in a non-decreasing order by effective length and let $\lambda_j = \ell_j/\ell_i$ for $j = 1,2,\ldots,t$. We have just shown that

$$x_0 \le \lambda_1 < f(\lambda_2) \le f(f(\lambda_3)) \le \cdots \le f^{(t-1)}(\lambda_t),$$

namely, $t - 1 \le f^*(\lambda_t) = O(f^*(\Delta))$. Thus, $|T| = |A| + |B| = O(1) + O(f^*(\Delta))$. ◀

Computability. Computability of our conflict graph construction is demonstrated through the notion of *inductive independence*. An n-vertex graph G is *k-inductive independent* if there is an ordering $v_1, v_2, \ldots, v_n$ of vertices such that for each v_i, the subgraph of G induced by the set $N_G(v_i) \cap \{v_i, v_{i+1}, \ldots, v_n\}$ has no independent set larger than k, where $N_G(v)$ denotes the neighborhood of vertex v. It is well known, e.g. [1, 35], that vertex coloring and MWISL problems are k-approximable in k-inductive independent graphs.

▶ **Theorem 7.** *Let f be a non-decreasing strongly sub-linear function with $f(x) \ge 40$ for all $x \ge 1$. For every set L, the graph $\mathcal{G}_f(L)$ is constant inductive independent.*

The proof is somewhat similar to that of Thm. 4. The inductive independence ordering non-decreasing order of links by length. With this in mind, the proof of Thm. 4 can be applied, with the following core difference: while in Thm. 4 the goal was, for a link i, to bound the number of *ρ-independent* links that have greater effective length and are f-adjacent with i, here we need to bound the number of *f-independent* links that have greater effective length and are f-adjacent with i.

Optimality of $O(\log\log\Delta)$-tightness. Here we show that the obtained tightness is essentially best possible, by demonstrating that every reasonable conflict graph formulation must incur an $O(\log\log\Delta)$ factor. We depart from some basic assumptions on conflict graphs. First, since the feasibility of a set of links is precisely determined by the values $\mathfrak{l}_i$ and d_{ij}, we assume that in a conflict graph, the adjacency of two links i, j is a predicate of variables $\mathfrak{l}_i, \mathfrak{l}_j, d_{ij}, d_{ji}$. Another basic observation is that the feasibility formula is scale-free w.r.t. those values; hence, we assume that so is a conflict graph formulation. This allows us to reduce the number of variables in the adjacency predicate: $\frac{\mathfrak{l}_{max}}{\mathfrak{l}_{min}}, \frac{d_{ij}}{\mathfrak{l}_{min}}, \frac{d_{ji}}{\mathfrak{l}_{min}}$, where $\mathfrak{l}_{min}$ and $\mathfrak{l}_{max}$ are the smaller and larger values of $\mathfrak{l}_i, \mathfrak{l}_j$, respectively. Our construction will consist of only *unit-length* links (i.e. $l_i = 1$) of mutual distance at least 3. In this case, we can further reduce the number of variables by noticing that in such instances, $d_{ij} = \Theta(d_{ji}) = \Theta(d(i,j))$. Thus, the conflict relation is essentially determined by two variables: $\frac{d(i,j)}{\mathfrak{l}_{min}}$ and $\frac{\mathfrak{l}_{max}}{\mathfrak{l}_{min}}$. By separating the variables, the conflict predicate boils down to a relation $\frac{d(i,j)}{\mathfrak{l}_{min}} > f(\frac{\mathfrak{l}_{max}}{\mathfrak{l}_{min}})$ for a function f. Note that this is similar to the conflict graph definition of [14], except that the lengths are replaced with effective lengths.

Let us show that the refinement property requires that $f(x) = \Omega(\sqrt{x})$ in such a graph. Let us fix a function $f : [1,\infty) \to [1,\infty)$. Let i, j be unit-length links with $\beta_j = 1$ and $\beta_i = X^\alpha > 1$, where X is a parameter. Assume further that the links i, j are placed on the plane so that $d(i,j) = 3f(X) = 3f(\mathfrak{l}_i/\mathfrak{l}_j)$, which means the links are f-independent. Thus, i, j must form a feasible set: $\frac{P(i)}{\mathfrak{l}_i^\alpha} > \frac{P(j)}{d_{ji}^\alpha}$ and $\frac{P(j)}{\mathfrak{l}_j^\alpha} > \frac{P(i)}{d_{ij}^\alpha}$. Multiplying these inequalities together and canceling $P(i)$ and $P(j)$ out, gives: $d_{ij}d_{ji} > \mathfrak{l}_i\mathfrak{l}_j = X$. This implies that we must have $d(i,j) = \Theta(\sqrt{d_{ij}d_{ji}}) = \Omega(\sqrt{X})$, which in turn implies that $f(X) = d(i,j)/3 = \Omega(\sqrt{X})$.

Now, a simple modification of the construction in [14, Thm. 9] gives a set S of unit-length links arranged on the line and with appropriately chosen thresholds β_i and distances $d(i,j)$, such that every two links are f-adjacent, but the whole set S is feasible. Such a construction can be done with the number of links $n = \Omega(f^*(\Delta))$, i.e. there is a *feasible* set of links that cannot be split in less than $\Omega(f^*(\Delta))$ f-independent subsets. Since $f(x) = \Omega(\sqrt{x})$, we have $f^*(x) = \Omega(\log\log x)$, which proves that the tightness must be at least $\Omega(\log\log\Delta)$.

5 Approximating Fixed-Rate Scheduling

We detail now the more classical problems that can be handled with our framework, starting with those involving fixed datarates. Intuitively, our framework can handle a problem if there is a correspondence between solutions in the physical model instance and solutions in the refinement graph. The refinement property ensures that the graph solutions map directly to feasible solutions in the physical model — we need to ensure a (approximate) correspondence in the other direction. We will argue that an optimal solution in the physical model has a counterpart in the graph instance, whose quality decreases only by the tightness factor k.

General Approximation Framework. Common scheduling-related optimization problems can be classified as *covering* or *packing*.

In covering problems, a feasible solution σ contains a (ordered) covering of the set L of links with feasible sets $\pi = \langle S_1, S_2, \ldots, S_t\rangle$ (i.e., $\cup_t S_t = L$), which we call *time slots*, and the objective is to minimize a function $f_\sigma(\pi)$ of the covering, which may also depend on other problem constraints.

In packing problems, a feasible solution σ contains a *fixed* number c of feasible sets (packing), $\eta = \langle S_1, S_2, \ldots, S_c\rangle$, not necessarily covering L, which we call *channels*, and the objective is to maximize a function $g_\sigma(\eta)$ of the packing.

Given a refinement $\mathcal{G}$ and a cover $\pi = \langle S_1, S_2, \ldots, S_t \rangle$ of L by feasible sets, we call another cover $\pi' = \langle S_1^1, \ldots S_1^{h_1}, S_2^1, \ldots, S_t^1, \ldots, S_t^{h_t} \rangle$, a refinement of π if $\langle S_i^1, \ldots S_i^{h_i} \rangle$ is a cover of S_i by *independent* sets in $\mathcal{G}$. Similarly, given a packing $\eta = \langle S_1, S_2, \ldots, S_c \rangle$, a refinement of η is another packing $\eta' = \langle S_1', S_2', \ldots, S_c' \rangle$, where $S_i' \subseteq S_i$ is an independent set in $\mathcal{G}$.

Formally, a covering problem is *refinable* if for every k-tight refinement $\mathcal{G}$ and a solution σ with cover π, there is a feasible solution σ' containing a refinement π' of π, and such that $f_\sigma(\pi) \geq \frac{f_{\sigma'}(\pi')}{k}$. A packing problem is *refinable* if for every k-tight refinement $\mathcal{G}$ and a solution σ with a packing $\eta = \langle S_1, S_2, \ldots, S_c \rangle$, there is a feasible solution σ' containing a refinement η' of η, and such that $g_\sigma(\eta) \leq k \cdot g_{\sigma'}(\eta')$.

▶ **Theorem 8.** *Let $\mathcal{G}$ be a k-tight refinement of the physical model. For every refinable problem, a ρ-approximation algorithm in $\mathcal{G}$ gives $k \cdot \rho$-approximation in the physical model.*

Thus, in order to obtain an approximation for a specific problem, it is sufficient to show that the problem is refinable: then the solution in a k-tight refinement gives a solution with an additional approximation factor k. Refinability requires the objective function of the problem to have certain linearity property. Examples of refinable covering problems include the ones where the objective function is the number of time slots or the sum of completion times (i.e. indices of time slots). Perhaps the simplest example of a refinable packing problem is the *maximal independent set of links* problem, where the objective is the size of the feasible set (i.e., there is only a single channel). Below, we apply the refinement framework to some important scheduling problems, which leads to $O(\log \log \Delta)$-approximation for all of them.

MWISL with Fixed Weights. Consider the MWISL problem, where the weights ω_i of links are fixed positive numbers. It is easy to see that this is a refinable packing problem, as the objective function – the sum of weights – is linear with respect to partition. Thus, since there is a constant factor approximation to MWISL in $\mathcal{G}(L)$ (by computability), it gives an $O(\log \log \Delta)$-approximation in the physical model (by Thm. 8).

Multi-Channel Selection. Given a natural number c – the number of channels – the goal is to select a maximum number of links that can be partitioned into c feasible subsets (a subset for each channel). Again, this is easily seen to be a refinable packing problem, as the objective function – the total number of links across all channels, is linear w.r.t. partitioning. A simple greedy algorithm gives constant factor approximation to multi-channel selection in constant-inductive independent graphs, which translates to an $O(\log \log \Delta)$-approximation in the physical model.

TDMA Scheduling. The goal is to partition the set L of links into the minimum number of feasible subsets. This is a covering problem, and the objective function is the number of slots, which is linear w.r.t. partitioning. A simple *first-fit* style greedy algorithm gives constant factor approximation to vertex coloring in constant inductive independent graphs, which gives an $O(\log \log \Delta)$-approximation to TDMA scheduling in the physical model.

Fractional Scheduling. This is a fractional variant of TDMA scheduling with an additional constraint of link demands. A *fractional schedule* for a set L of links is a collection of feasible sets with rational values $\mathcal{S} = \{(I_k, t_k) : k = 1, 2 \ldots, q\} \subseteq \mathcal{E}_\mathcal{P} \times \mathbb{R}_+$, where $\mathcal{E}_\mathcal{P}$ is the set of all feasible subsets of L. The sum $\sum_{k=1}^{q} t_k$ is the *length* of the schedule $\mathcal{S}$. The *link capacity vector* $c_\mathcal{S} : L \to \mathbb{R}_+$ associated with the schedule $\mathcal{S}$ is given by $c_\mathcal{S}(i) = \sum_{(I,t) \in \mathcal{S} : I \ni i} t$.

Essentially, the link capacity shows how much scheduling time each link gets. Finally, a *link demand vector* $d : L \to \mathbb{R}_+$ indicates how much scheduling time each link needs.

The *fractional scheduling problem* is a covering type problem, where given a demand vector d, the goal is to compute a minimum length schedule that serves the demands d, namely, for each link $i \in L$, $c_{\mathcal{S}}(i) \geq d(i)$. Since the cost function $\sum_{k=1}^{q} t_k$ is again linear w.r.t. partitioning of a schedule, it is readily checked that the fractional scheduling problem is also refinable. A simple greedy algorithm presented in [31] achieves constant factor approximation for fractional scheduling in constant inductive independent graphs. This gives an $O(\log\log\Delta)$-approximation in the physical model.

Joint Routing and Scheduling. Consider an ordered set of p source-destination node pairs (multihop communication requests) (u_i, v_i), $i = 1, 2, \ldots, p$, with associated weights/utilities $\omega_i > 0$, in a multihop network given by a directed graph G, where the *edges* of the graph are the transmission links. Let $\mathcal{P}_i$ denote the set of directed (u_i, v_i) paths in G and let $\mathcal{P} = \cup_i \mathcal{P}_i$. Then a *path flow* for the given set of requests is a set $\mathcal{F} = \{(P_k, \delta_k) : k = 1, 2, \ldots\} \subseteq \mathcal{P} \times \mathbb{R}_+$. The *link flow vector* $f_{\mathcal{F}}$ corresponding to path flow $\mathcal{F}$, with $f_{\mathcal{F}}(i) = \sum_{(P,\delta)\in\mathcal{F}:P\ni i} \delta$ for each link i, shows the flow along each link.

The *multiflow routing and scheduling problem* is a covering problem, where given source-destination pairs with associated utilities, the goal is to find a path flow $\mathcal{F}$ together with a fractional link schedule $\mathcal{S}$ of length 1, such that[2] for each link i, the link flow is at most the link capacity provided by the schedule, $f_{\mathcal{F}}(i) \leq c_{\mathcal{S}}(i)$, and the *flow value*

$$W = \sum_{i=1}^{p} \omega_i \cdot \sum_{(P_k,\delta_k)\in\mathcal{F}, P_k\in\mathcal{P}_i} \delta_k$$

is maximized. Let us verify that this problem is also refinable. Consider a feasible solution in (the physical model) that consists of a path flow $\mathcal{F} = \{(P_k, \delta_k) : k = 1, 2, \ldots\}$ and a schedule $\mathcal{S} = \{(I_k, t_k) : k = 1, 2, \ldots\}$ of length $\sum_{k\geq 1} t_k = 1$, such that $f_{\mathcal{F}}(i) \leq c_{\mathcal{S}}(i)$. As observed in the previous section, the schedule $\mathcal{S}$ can be refined into a schedule $\mathcal{S}' = \{(I_k^s, t_k)\}_{k,s}$ in $\mathcal{G}(L)$, where $\mathcal{S}'$ serves the same demand vector as $\mathcal{S}$ does, and $\mathcal{S}'$ has length at most $K = O(\log\log\Delta)$ times more than the length of $\mathcal{S}$. Now we normalize the refined schedule to have length 1. Then, the following modified path flow $\mathcal{F}' = \{(P_k, \delta_k/K) : k = 1, 2, \ldots\}$ together with the new schedule will be feasible in $\mathcal{G}(L)$, as all link demands will be served. Moreover, the value of $\mathcal{F}'$ is at most K times that of $\mathcal{F}$. Hence, the problem is refinable.

Thus, applying the constant factor approximation algorithm of [32] for constant inductive independent conflict graphs (the result holds with unit utilities) gives an $O(\log\log\Delta)$-approximation for multiflow routing and scheduling problem in the physical model. It should also be noted that the fractional scheduling and routing and scheduling problems can be reduced to the Mwisl problem using linear programming techniques (described e.g. in [17]), as it was shown in [30]. We will further discuss this in Section 6.

Extensions to Multi-Channel Multi-Antenna Settings. All problems above may be naturally generalized to the case when there are several channels (e.g. frequency bands) available and moreover, wireless nodes are equipped with multiple antennas and can work in different channels simultaneously. We denote the setting with multiple antennas/channels as *MC-MA*.

[2] Essentially, the schedule here gives a probability distribution over the feasible sets of links.

It is easy to show that our refinement framework can be extended to MC-MA with very little change. Assume that each node u is equipped with $a(u)$ antennas numbered from 1 to $a(u)$ and can use a set $\mathcal{C}(u)$ of channels. Consider a link i that corresponds to the pair of nodes s_i and r_i. There are $a(s_i)a(r_i)|\mathcal{C}(s_i) \cap \mathcal{C}(r_i)|$ *virtual* links corresponding to each selection of an antenna of the sender node s_i, an antenna of receiver node r_i and a channel $c \in \mathcal{C}(s_i) \cap \mathcal{C}(r_i)$ available to both nodes. Thus a virtual link is described by the tuple (i, a_s, a_r, c), where a_s (a_r) denotes the antenna index at s_i (r_i, respectively), and c denotes the channel. We call link i *the original* of its virtual links. Note that the formulation above can easily be generalized to the case where certain antennas don't work in certain channels, e.g., due to multi-path fading.

A set of (virtual) links S is feasible in MC-MA if and only if no two links in S share an antenna (i.e., they do not use the same antenna of the same node), and for each channel c, the set of originals of links in S using channel c is feasible in the physical model. Then, an $O(\log\log\Delta)$-tight refinement for the MC-MA physical model by a conflict graph can be found by a simple extension of the existing refinement for the single channel case to the virtual links. This implies, in particular, that all scheduling problems considered in the previous sections can also be approximated in the MC-MA setting within an approximation factor $O(\log\log\Delta)$, as the corresponding approximations for the conflict graph hold with MC-MA [32].

6 Rate Control and Scheduling

The most important application of efficient approximation algorithms for scheduling problems with different thresholds is the application to scheduling with rate control. This is achieved first by obtaining a double-logarithmic approximation to Mwisl with rate control. This will then lead to similar approximations for fractional scheduling and joint routing and scheduling problems.

MWISL with Rate Control. By Shannon's theorem, given a set S of links simultaneously transmitting in the same channel, the transmission rate $r(S, i)$ of a link i is a function of $SIR(S, i)$. Thus, we consider the Mwisl problem where each link i has an associated *utility function* $u^i : \mathbb{R}_+ \to \mathbb{R}_+$, and the weight of link i is the value of u^i at $SIR(S, i)$ if link i is selected in the set, and 0 otherwise. As before, the goal is, given the links with utility functions, to find a subset S that maximizes the total weight $\sum_{i\in S} u^i(r(S, i))$. We assume that $u^i(SIR(S, i)) = 0$ if $SIR(S, i) < 1$.

An $O(\log n)$-approximation for this variant of Mwisl has been obtained in [23]. We show that this can be replaced with $O(\log\log\Delta')$, where $\Delta'(L) = \max_{i,j\in L} \frac{u^i_{max} l_i}{u^j_{min} l_j}$ and u^i_{min}, u^i_{max} are the minimum and maximum possible utility values for the given instance and link. This is achieved by reducing the problem to Mwisl in an extended instance.

Let us fix a utility function u. First, assume that the possible set of weights for each link is a discrete set $u_{min} = u_1 < u_2 < \cdots < u_t = u_{max}$. Then, we can replace each link i with t copies $i_1, i_2, \cdots, i_t$ with different thresholds and fixed weights, where $\omega_{i_k} = u_k$ and $\beta_{i_k} = \min\{x : u^{i_k}(x) \geq u_k\}$ if $\beta_{i_k} \geq 1$ and $\omega_{i_k} = 0$ otherwise. Now, the problem becomes a Mwisl problem for the modified instance L' with link replicas and fixed weights. Observe that no feasible set in L' contains more than a single copy of the same link, as the copies occupy the same geometric place, implying that each feasible set of the extended instance corresponds to a feasible set of the original instance, with an obvious transformation. The effective length diversity of the extended instance is $\Delta(L') = \Delta'(L)$. Thus, using the approximation

algorithm for the fixed rate MWISL problem, we obtain an $O(\log\log\Delta'(L))$-approximation for MWISL with rate control.

For the case when the number of possible utility values is too large or the set is continuous, a standard trick can be applied. Let u^i_{max}, u^i_{min} be as before. The extended instance L' is constructed by replacing each link i with $O(\log u^i_{max}/u^i_{min})$ copies $i_1, i_2, \ldots$ of itself and assigning each replica i_k weight $\omega_k = 2^{k-1}$ and threshold $\beta_k = \min\{x : 2^{k-1} \le u^i(x) \le 2^k\}$ if $\beta_k \ge 1$ and let $\omega_k = 0$ otherwise. It is easy to see that the optimum value of MWISL with fixed rates in L' is again an $O(\log\log\Delta'(L))$-approximation to MWISL with rate control.

If the value $\log u^i_{max}/u^i_{min}$ is still too large, it may be inefficient to have $O(\log u^i_{max}/u^i_{min})$ copies for each link. It is another standard observation that only the last $O(\log n)$ copies of each link really matter, as restricting to only those links degrades approximation by a factor of at most 2.

Fractional Scheduling with Rate Control. In this formulation, we redefine a fractional schedule to be a set $\mathcal{S} = \{(I_k, t_k) : k = 1, 2 \ldots, q\} \subseteq 2^L \times \mathbb{R}_+$, namely, I_k are arbitrary subsets, rather than independent ones. We redefine the link capacity vector $\hat{c}_\mathcal{S}$ to incorporate the rates as follows:

$$\hat{c}_\mathcal{S}(i) = \sum_{(I,t)\in\mathcal{S}: I\ni i} t \cdot r(i, I). \tag{2}$$

The *fractional scheduling with rate control* problem is to find a minimum length schedule $\mathcal{S}$ that serves a given demand vector d, namely, such that for each link $i \in L$, $\hat{c}_\mathcal{S}(i) \ge d(i)$.

The problem can be formulated as an exponential size linear program LP_1, as follows.

$$\min \sum_{I\subseteq L} t_I \text{ subject to } \sum_{I\subseteq L: I\ni i} t_I \cdot r(i, I) \ge d(i) \qquad \forall i \in L$$
$$t_I \ge 0 \qquad \forall I \subseteq L$$

The dual program LP_2 looks as follows:

$$\max \sum_{i\in L} d(i) y_i \text{ subject to } \sum_{i\in I} y_i \cdot r(i, I) \ge 1 \qquad \forall I \subseteq L$$
$$y_i \ge 0 \qquad \forall i \in L$$

As [17, Thm. 5.1] states, if there is an approximation algorithm that finds a set $\hat{I}$ such that $\sum_{i\in\hat{I}} y_i r(i, \hat{I}) \ge \frac{1}{a} \max_{I\subseteq L} \sum_{i\in I} y_i r(i, I)$, then there is an a-approximation algorithm for LP_1, where the former algorithm acts as an approximate separation oracle for LP_1. But this auxiliary problem is simply a special case of the MWISL with rate control, which we can approximate within a double-logarithmic factor. Thus, there is an approximation preserving reduction from the fractional scheduling with rate control to MWISL with rate control. By the obtained approximation for MWISL, we obtain an $O(\log\log\Delta')$-approximation for fractional scheduling with rate control.

Routing, Scheduling and Rate Control. The rate-control variant of the routing and scheduling problem is formulated in the same way as for the fixed rate setting, with the only modified constraint being the capacity constraints, which, instead of the link capacity vector $c_\mathcal{S}$, now use the modified variant $\hat{c}_\mathcal{S}$ that incorporates the link rates (see the definition in (2)).

This problem can also be reduced to MWISL with rate control, using similar methods as for the fractional scheduling problem. The reduction is nearly identical to the reduction of fixed rate versions of these problems to MWISL, presented in [30, Thm. 4.1].

Thus, we can conclude that there is an $O(\log \log \Delta')$-approximation algorithm for joint routing, scheduling and rate control that uses MWISL with rate control as a subroutine.

References

1. Karhan Akcoglu, James Aspnes, Bhaskar DasGupta, and Ming-Yang Kao. Opportunity cost algorithms for combinatorial auctions. *CoRR*, cs.CE/0010031, 2000.
2. Mahmoud Al-Ayyoub and Himanshu Gupta. Joint routing, channel assignment, and scheduling for throughput maximization in general interference models. *IEEE Trans. Mob. Comput.*, 9(4):553–565, 2010. doi:10.1109/TMC.2009.144.
3. Jørgen Bang-Jensen and Magnús M. Halldórsson. Vertex coloring edge-weighted digraphs. *Inf. Process. Lett.*, 115(10):791–796, 2015. doi:10.1016/j.ipl.2015.05.007.
4. Deepti Chafekar, V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. Cross-layer latency minimization in wireless networks with SINR constraints. In *Proceedings of the 8th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2007, Montreal, Quebec, Canada, September 9-14, 2007*, pages 110–119, 2007. doi:10.1145/1288107.1288123.
5. Michael Dinitz. Distributed algorithms for approximating wireless network capacity. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 1397–1405, 2010. doi:10.1109/INFCOM.2010.5461905.
6. Alexander Fanghänel, Thomas Kesselheim, and Berthold Vöcking. Improved algorithms for latency minimization in wireless networks. *Theor. Comput. Sci.*, 412(24):2657–2667, 2011. doi:10.1016/j.tcs.2010.05.004.
7. Liqun Fu, Soung Chang Liew, and Jianwei Huang. Power controlled scheduling with consecutive transmission constraints: Complexity analysis and algorithm design. In *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil*, pages 1530–1538, 2009. doi:10.1109/INFCOM.2009.5062070.
8. Olga Goussevskaia, Yvonne Anne Oswald, and Roger Wattenhofer. Complexity in geometric SINR. In *Proceedings of the 8th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2007, Montreal, Quebec, Canada, September 9-14, 2007*, pages 100–109, 2007. doi:10.1145/1288107.1288122.
9. Olga Goussevskaia, Luiz Filipe M. Vieira, and Marcos Augusto M. Vieira. Wireless scheduling with multiple data rates: From physical interference to disk graphs. *Computer Networks*, 106:64–76, 2016. doi:10.1016/j.comnet.2016.06.016.
10. Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. Information Theory*, 46(2):388–404, 2000. doi:10.1109/18.825799.
11. Magnús M. Halldórsson. Wireless scheduling with power control. *ACM Trans. Algorithms*, 9(1):7:1–7:20, 2012. doi:10.1145/2390176.2390183.
12. Magnús M. Halldórsson and Pradipta Mitra. Wireless capacity with oblivious power in general metrics. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1538–1548, 2011. doi:10.1137/1.9781611973082.119.
13. Magnús M. Halldórsson and Pradipta Mitra. Wireless capacity and admission control in cognitive radio. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pages 855–863, 2012. doi:10.1109/INFCOM.2012.6195834.
14. Magnús M. Halldórsson and Tigran Tonoyan. How well can graphs represent wireless interference? In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 635–644, 2015. doi:10.1145/2746539.2746585.

15 Magnús M. Halldórsson and Tigran Tonoyan. The price of local power control in wireless scheduling. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 529–542, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.529.

16 Juha Heinonen. *Lectures on Analysis on Metric Spaces*. Springer, 1. edition, 2000.

17 Klaus Jansen. Approximate strong separation with application in fractional graph coloring and preemptive scheduling. *Theor. Comput. Sci.*, 302(1-3):239–256, 2003. doi:10.1016/S0304-3975(02)00829-0.

18 Libin Jiang, Mathieu Leconte, Jian Ni, R. Srikant, and Jean C. Walrand. Fast mixing of parallel glauber dynamics and low-delay CSMA scheduling. *IEEE Trans. Information Theory*, 58(10):6541–6555, 2012. doi:10.1109/TIT.2012.2204032.

19 Libin Jiang and Jean C. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Trans. Netw.*, 18(3):960–972, 2010. doi:10.1109/TNET.2009.2035046.

20 Changhee Joo, Xiaojun Lin, and Ness B. Shroff. Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks. *IEEE/ACM Trans. Netw.*, 17(4):1132–1145, 2009. doi:10.1145/1618562.1618572.

21 Bastian Katz, Markus Völker, and Dorothea Wagner. Energy efficient scheduling with power control for wireless networks. In *8th International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt 2010), May 31 - June 4, 2010, University of Avignon, Avignon, France*, pages 160–169, 2010.

22 Thomas Kesselheim. A constant-factor approximation for wireless capacity maximization with power control in the SINR model. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1549–1559, 2011. doi:10.1137/1.9781611973082.120.

23 Thomas Kesselheim. Approximation algorithms for wireless link scheduling with flexible data rates. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, pages 659–670, 2012. doi:10.1007/978-3-642-33090-2_57.

24 Henry Lin and Frans Schalekamp. On the complexity of the minimum latency scheduling problem on the euclidean plane. *CoRR*, abs/1203.2725, 2012.

25 Xiaojun Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, volume 2, pages 1484–1489 Vol.2, Dec 2004. doi:10.1109/CDC.2004.1430253.

26 Thomas Moscibroda and Roger Wattenhofer. The complexity of connectivity in wireless networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain*, 2006. doi:10.1109/INFOCOM.2006.23.

27 Devavrat Shah, David N. C. Tse, and John N. Tsitsiklis. Hardness of low delay network scheduling. *IEEE Trans. Information Theory*, 57(12):7810–7817, 2011. doi:10.1109/TIT.2011.2168897.

28 L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, Dec 1992. doi:10.1109/9.182479.

29 Tigran Tonoyan. On some bounds on the optimum schedule length in the SINR model. In *Algorithms for Sensor Systems, 8th International Symposium on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities, ALGOSENSORS 2012, Ljubljana, Slovenia, September 13-14, 2012. Revised Selected Papers*, pages 120–131, 2012. doi:10.1007/978-3-642-36092-3_14.

30 Peng-Jun Wan. Multiflows in multihop wireless networks. In *Proceedings of the 10th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2009, New Orleans, LA, USA, May 18-21, 2009*, pages 85–94, 2009. doi:10.1145/1530748.1530761.

31 Peng-Jun Wan, Xiaohua Jia, Guojun Dai, Hongwei Du, Zhiguo Wan, and Ophir Frieder. Scalable algorithms for wireless link schedulings in multi-channel multi-radio wireless networks. In *Proceedings of the IEEE INFOCOM 2013, Turin, Italy, April 14-19, 2013*, pages 2121–2129, 2013. doi:10.1109/INFCOM.2013.6567014.

32 Peng-Jun Wan, Zhu Wang, Lei Wang, Zhiguo Wan, and Sai Ji. From least interference-cost paths to maximum (concurrent) multiflow in MC-MR wireless networks. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 334–342, 2014. doi:10.1109/INFOCOM.2014.6847955.

33 Lixin Wang, C. P. Abubucker, William F. Lawless, and Anthony J. Baker. A constant-approximation for maximum weight independent set of links under the SINR model. In *Seventh International Conference on Mobile Ad-hoc and Sensor Networks, MSN 2011, Beijing, China, December 16-18, 2011*, pages 9–14, 2011. doi:10.1109/MSN.2011.1.

34 Xinzhou Wu, R. Srikant, and James R. Perkins. Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. *IEEE Trans. Mob. Comput.*, 6(6):595–605, 2007. doi:10.1109/TMC.2007.1061.

35 Yuli Ye and Allan Borodin. Elimination graphs. *ACM Trans. Algorithms*, 8(2):14:1–14:23, 2012. doi:10.1145/2151171.2151177.

Streaming Communication Protocols*†

Lucas Boczkowski[1], Iordanis Kerenidis[2], and Frédéric Magniez[3]

1 **CNRS, IRIF, Univ Paris Diderot, Paris, France**
`lucas.boczkowski@irif.fr`
2 **CNRS, IRIF, Univ Paris Diderot, Paris, France**
`iordanis.kerenidis@irif.fr`
3 **CNRS, IRIF, Univ Paris Diderot, Paris, France**
`frederic.magniez@irif.fr`

Abstract

We define the Streaming Communication model that combines the main aspects of communication complexity and streaming. Input arrives as a stream, spread between several agents across a network. Each agent has a bounded memory, which can be updated upon receiving a new bit, or a message from another agent. We provide tight tradeoffs between the necessary resources, i.e. communication between agents and memory, for some of the canonical problems from communication complexity by proving a strong general lower bound technique. Second, we analyze the Approximate Matching problem and show that the complexity of this problem (i.e. the achievable approximation ratio) in the one-way variant of our model is strictly different both from the streaming complexity and the one-way communication complexity thereof.

1998 ACM Subject Classification C.2.2 Network Protocols

Keywords and phrases Networks, Communication complexity, Streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.130

1 Introduction

In the last decade we have witnessed a big shift in the way data is produced and computation is performed. First, we now have to deal with enormous amounts of data that we cannot even store in memory (internet traffic, CERN experiments, space expeditions). Second, computations do not happen in a single processor or machine, but with multi-core processors and multiple machines in cloud architectures. All these real-world changes necessitate that we revisit and extend our models and tools for studying the efficiency and hardness of computational problems.

Imagine the following situation: some input is spread across a network. The agents want to compute some function f which depends on everybody's input.This is an archetypal problem of Communication Complexity (CC) [29], which offers a way to estimate the number of bits that need to be exchanged, under various settings, in order to achieve that goal. There are many different CC models, depending whether the agents can speak directly between them or through a referee, and whether they can use multiple rounds of communication or just a single one. Communication complexity has found a variety of applications both in networks and distributed computing but also in other areas of theoretical computer science, including, circuit lower bounds, fomulae size, VLSI design, etc. All these communication

* Full version available at `https://arxiv.org/abs/1609.07059`.
† This work has been partially supported by the ERC project QCC and the French ANR Blanc project RDAM.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 130; pp. 130:1–130:14


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

models however have a feature in common. They assume the agents are computationally unbounded and that the input is delivered all at once.

In a distributed context as that of sensor networks, not only are there several computing agents, the input might not even be given all at once. The *streaming model* [1] has been defined precisely to capture the fact that the input of an agent is so big it cannot be stored or read several times. Instead it comes bit by bit. Some function of the stream needs to be computed, but the available space is not big enough to store the entire input. The streaming model has been extensively studied in recent years with a plethora of interesting upper and lower bounds on the necessary memory to solve specific streaming problems [27]. More recently, the turnstile model has received a lot of attention. In this model, streams are made of both insertions and deletions, and the function to be computed depends on the remaining elements (and eventually their respective frequencies). Indeed, any streaming algorithm in this model can be turned into an algorithm based solely on the updates of linear sketches [25].

1.1 The Streaming Communication model.

We would like to combine the two above mentioned models to include both that inputs are distributed among different agents and also are coming as streams at each agent. Each agent is given a bounded memory to store what she sees. We refer to this extension as the *Streaming Communication (SC) model*. Even though communication arguments have often been invoked in proving lower bounds for regular streaming models, as in the seminal work of [1, 3], this model has not been rigorously defined previously, in spite of its theoretical appeal and relevance for actual communication networks. More formally, in the SC model consider two agents, Alice and Bob, want to compute some function f that depends on inputs (x, y) that are respectively distributed to each agent, x to Alice and y to Bob. Both inputs arrive as data streams and each agent has a bounded memory of a given size S. Agents may or may not speak every time they receive a bit. They can also update their memory based on the previous bit they read, the previous message they received and of course the actual content of their memory.

Additionally to the memory size S of Alice and Bob, the other relevant parameters we consider are the number R of communication rounds and the number T of bits in the full transcript (the concatenation of all messages). In the *one-way* SC model, there is only a single message from Alice to Bob at the end of the streams. Observe that we do not bound the size of each message, since we show that those can always be assumed to be of at most $S + 1$ bits (**Proposition 5**).

1.2 Related models

Before we present our results in the streaming communication model, we review some related works in the communication and streaming models. As explained previously, our goal is to provide rigorous tradeoffs between the two resources: memory storage and communication between the players, in a model where inputs are coming as streams.

The most relevant work is [15, 16]. There, two parties receive two streams and at the end of the streams each party sends their workspace to a referee which uses both workspaces to compute some function of the union of the two streams. In this model, we can also see elements both from streaming algorithms and communication complexity, albeit of the restricted form of simultaneous message passing. Here we provide a more general framework for communication and we look at a much wider variety of problems and protocols.

One of the powerful and often used techniques in streaming algorithms is linear sketches, which naturally provide very simple SC protocols even in the one-way setting, by just combining the linear sketches at the end of the protocol. In fact, in the turnstile model, when streams are made of sequences of insertions and deletions and the function to be computed is a function of the remaining elements (and eventually their respective frequencies), any streaming algorithm can be turned into an algorithm based solely on the updates of linear sketches [25]. Hence, here we focus on other stream models, such as the one of insertion only, which are more challenging in the context of SC protocols.

In the distributed streaming functional monitoring setup initially proposed by [11], k servers receiving a stream have to allow a coordinator to continuously monitor a given quantity (see also [10, 9, 13] for earlier works in the database community on the monitoring topic). Several follow-up works studied this model (see e.g. [8, 26, 18, 12, 28]). These all focus on communication and do not consider *both* resources, memory storage and communication simultaneously. They can be viewed as extending [15, 16] with greater number of players. The communication model however is still restricted to simultaneous messages to a referee.

Another line of work studies bounded-memory versions of communication [24, 5, 7]. Several models have been proposed that share the same structure. The input is given all at once, but the players only have bounded space to store the conversation while further restrictions can be placed on the algorithms used by the players, for example to be straight line programs [24], or branching programs [5].

Last, [14] studies another model that deals with distributed parallel streaming platforms, where now, the stream arrives in parallel and arbitrarily partitioned to a set of different agents that communicate in order to solve the task.

1.3 Our results

As a first step, we restrict ourselves in this work to the 2-player case. Our first results, detailed in Section 2.4, show connections between our new model and its two parent models, Communication Complexity and Streaming. We show that the total transcript size T and the product RS (number of rounds times memory size) are both lower bounded by the communication complexity $C(f)$ of the f we wish to compute, up to logarithmic factors (**Proposition 6**). Those factors come from an inherent notion of clock in our SC model.

The comparison with streaming algorithms is more subtle. Since the i-th bit of Alice's input arrives at the same time as the i-th bit of Bob's input, the correct comparison is with a single streaming model where the stream is the one we get by interleaving Alice's and Bob's stream. We denote by $\mathcal{S}^{\mathbf{int}}(f)$ the memory required by a streaming algorithm for computing a function f, when the two streams are interleaved in a single stream. We first observe than interleaving streams instead of concatenating them can lead to an exponential blow up (**Theorem 8**). Then we show that $\mathcal{S}^{\mathbf{int}}(f)$ is a lower bound on twice the memory size S of players (**Proposition 9**).

Then it is natural to ask if there is always a polynomial relation between, on one hand, the parameters of a protocol in the SC model (S, R and T), and on the other hand, the communication complexity $C(f)$ (randomized or deterministic) of the function when the input is all given in the beginning and the memory $\mathcal{S}^{\mathbf{int}}(f)$ necessary in the single stream model. We show that this is not true in general by providing an example for which $\mathcal{S}^{\mathbf{int}}(f) = C(f) = O(\log n)$ but $R \cdot S = \Omega(n)$, when $S = \Omega(\log n)$ (**Theorem 10**). This implies that the SC complexity of a function f may not be immediately derived neither by its communication complexity nor by its streaming complexity. This is one of the main reasons why our model is interesting and necessitates novel techniques for its study.

The first of our two main results is a general technique for proving tradeoffs between memory and communication in our model. The smaller the memory, the more frequent communication has to be. For instance, one expects that for functions whose communication complexity is n, i.e. where all bits are necessary, players with a memory of size S *have to* speak at least every S rounds (either deterministic or randomized), since if they remain silent for more than S rounds, they start to lose information about their input. More precisely, any function f that can be written as $f(x,y) = G(g_1(x^1,y^1), g_2(x^2,y^2), \ldots, g_L(x^L,y^L))$, where G is a function satisfying some assumptions. Then, any randomized protocol computing f must have $R \cdot S = \Omega(\sum_{\ell \in L} C(g_\ell))$ (**Theorem 13**). We can apply our theorem to many canonical communication functions, including $IP_n, DISJ_n$ or $TRIBES_n$, and show that any protocol satisfies $R \cdot S = \Omega(n)$ (**Theorem 14**).

In Section 4, we study problems arising in the context of graph streaming. We work in the insert only model, meaning that the graph is presented as a stream of its edges in an arbitrary order. Indeed, as opposed to the turnstile model, where any algorithm can be turned into a linear sketches based on [25], the situation is much more intriguing for problems where linear sketches are not used. In particular, in the context of streaming algorithms for graph problems, *Approximate Matching* has been extensively studied, and its streaming complexity is still unknown. Given a stream of edges (in an arbitrary order) of an n-vertex graph G and some space restriction, the goal is to output a collection of edges from G forming a matching, as big as possible in G. The matching size estimation is a different and somehow easier problem [21].

It is known that any streaming algorithm for Approximate Matching using $\widetilde{O}(n)$ memory cannot achieve a ratio better than $\frac{e}{e-1}$ [20], whereas the best known algorithm is a simple greedy algorithm which provides a 2-approximation. In the one-way CC model, without memory constraints, it has been also showed that a $\frac{3}{2}$-approximation is the tight bound when Alice's message is restricted to $\widetilde{O}(n)$ bits [17]. Both these works use in a clever way the so-called Ruzsa-Szemerédi graphs.

We study both the general SC model and its one-way variant. Our main bounds are for the one-way variant, the weaker model combining the restrictions of both CC and streaming models: we show a lower bound of $\frac{e+1}{e-1} \approx 2.16$ for the approximation factor unless $S = n^{1+\Omega(\frac{1}{\log \log n})}$ (**Corollary 17**), which is strictly higher than both the single stream lower bound of $\frac{e}{e-1} \approx 1.58$ with same space constraints and the one-way communication lower bound of 1.5. We also provide a one-way SC protocol achieving an approximation ratio of 3 with the same space constraints (**Theorem 16**), thus leaving as an open question the optimal approximation ratio. Moreover, we show that how often the players communicate makes a big difference, namely we show how to implement the simple greedy algorithm when Alice and Bob can communicate during the protocol that provides a ratio of 2, strictly better than our lower bound for the one-way SC model.

Let us emphasize that all previous lower bounds, including the ones in the turnstile models [2, 22], do not readily apply to the one-way SC model for the Approximate Matching problem. However, our main lower bound in the one-way SC model uses as a black-box the hard distributions of graph streams of [17, 20]. Therefore, further improvements in the streaming context may lead to improvements in our model. Given a hard distribution μ of graphs for the approximate matching for streaming algorithms, we show how to extend this distribution to produce a hard distribution μ_2 in our one-way SC model (**Theorem 21**).

2 The streaming communication model

We provide some background on communication complexity and streaming and then, we define our model and describe some initial results.

2.1 Communication Complexity

We start by reviewing some results in the usual models of communication complexity (CC), defined by Yao [29]. For more details about the communication complexity model, please refer to [23]. In the communication complexity models, generically denoted by CC, players aim at computing some function which depends on their disjoint inputs, by communicating. Each player determines her message based on previous messages and her input. The goal is to minimize the total length of the protocol transcript.

In the randomized case, we will allow the players to share public randomness. Allowing for public randomness makes our lower bounds stronger, while the protocols we provide will be deterministic. We will also consider the expected, rather than maximal, length of transcripts and define the average randomized communication complexity of a function.

▶ **Definition 1.** For a given protocol Π, we denote by $\Pi(x, y, r)$ the transcript with inputs x, y and public randomness r. The worst case (resp. expected) communication complexity of a function f with error ε is defined as $C_\varepsilon(f) = \min_\Pi \max_{x,y} \max_r |\Pi(x, y, r)|$ and $C_\varepsilon^{avg}(f) = \min_\Pi \max_{x,y} \mathbb{E}_r(|\Pi(x, y, r)|)$, where the minimum is taken over protocols computing f with error ε, and the expectation on the second line is with respect to the randomness r used in Π.

The following proposition relates the average and worst case randomized communication complexities.

▶ **Proposition 2** ([23]). *For any $\varepsilon, \delta > 0$, it holds that, $\delta \cdot C_{\varepsilon+\delta}(f) \leq C_\varepsilon^{avg}(f) \leq C_\varepsilon(f)$.*

Some of the canonical functions studied in communication complexity are the equality problem, denoted EQ_n where the players output 1 iff their inputs $x, y \in \{0, 1\}^n$ are equal, the disjointness problem, denoted $DISJ_n$ where the goal is to check whether the n-bit strings interpreted as sets intersect or not, and the inner product problem IP_n where the players need to output the inner product of their inputs modulo 2.

The functions $DISJ_n$ and IP_n are "hard" functions for CC, in the sense that almost all the input must be sent even when we allow for randomization, error and expected length. The following two bounds, which we will need later, can be derived for example from [4], where the notion of information cost is used.

▶ **Theorem 3.** *Any protocol for $DISJ_n$ or IP_n with error $1/2 - \varepsilon$ has communication complexity $\Omega(\varepsilon^2 n)$.*

2.2 Streaming algorithms

In the streaming model, the input comes as a stream to an algorithm whose task is to compute some function of the stream while using only a limited amount of memory and making a single or a few passes through the input stream. See [27] for a general introduction to the topic. If possible, the updates should also be fast. It was defined in the seminal work of [1] where the authors provided upper and lower bounds for computing some stream statistics. Since then, a plethora of results have appeared for computing statistics of the stream, as well as for graph theoretic problems. For the graph problems, we will assume

that the graph is revealed to the algorithm as a stream, one edge at a time. In the more recent turnstile model, streams are made of both insertions and deletions, and the goal is to compute some function that depends only the remaining elements (and eventually their respective frequencies). As we have said, any problem in the turnstile model can be solved via linear sketches [25].

2.3 The new model of Streaming Communication protocols

We show how to extend the original model of communication complexity to account for streaming inputs. In the Streaming Communication SC model we consider that the inputs x, y are not given all at once to the two players Alice and Bob but rather come as a stream. Moreover each player only has limited storage, S bits of memory. In the randomized case, the players also have access to a shared random bit string r which may be infinite. They may use as many coins as they like from these strings.

A *protocol* Π in the streaming communication model is specified by four functions $\Phi^A, \Phi^B, \Psi^A, \Psi^B$. Each time slot i is divided in two phases:

1. Each party receives a message from the other party (m_i^B and m_i^A resp.) and updates their memory (that was in state σ_i^A and σ_i^B resp.) according to the function Φ^A and Φ^B resp. This function also depends and the shared random string r, which is not restricted in size.
2. Messages m_{i+1}^A and m_{i+1}^B are produced using the functions Ψ^A and Ψ^B resp., that depend on the current memory states σ_{i+1}^A and σ_{i+1}^B resp., the newly read input bit, and the randomness r. The messages might be empty and they could also be arbitrarily big in principle, though we will see in Proposition 5 that their size can be assumed to be $S+1$ without loss of generality.

$\sigma_{i+1}^{A/B} := \Phi^{A/B}(m_i^{B/A}, \sigma_i^{A/B}, r)$ and $m_{i+1}^{A/B} := \Psi^{A/B}(\sigma_{i+1}^{A/B}, x_{i+1}, r)$. Moreover, we assume that the streams end with a special EOF symbol and that once the streams are finished, the players only get one last round of communication, and then they have to output something.

▶ **Definition 4** (SC protocols)**.** An SC protocol Π uses S bits of memory, R rounds, and T bits when

1. The memory size of each player is at most S bits;
2. The (expected) number of time slots where either $m_i^A \neq \emptyset$ or $m_i^B \neq \emptyset$ is at most R;
3. The (expected) size of all exchanged messages is at most T bits.

The expectation is over the randomness of the protocol and worst-case over the inputs. An SC protocol is said to be *one-way* if there is a single message from Alice to Bob after the streams have been received, and only Bob computes the function.

Note, that our model carries an implicit notion of time due to the players reading their streams synchronously, and hence, the ability to send empty messages can be used to reduce communication [19]. However the gain is only logarithmic in the number of available time slots (see Section 2.4). We could have avoided such extra power, by defining a model where agents know when they should speak or read a bit, based on the previous messages they received and their memory content. Nevertheless, we opted for our model, as it is simpler to state and the necessary resources do not change by more than a factor logarithmic in the input size.

When we prove lower bounds or communication-memory tradeoffs, we do not consider the complexity of $\Phi^{A/B}$. These functions could be of arbitrarily high complexity. To make things simple, we assume they are the same functions for every round $i \in [n]$, but they can

depend on n. This framework captures the streaming model as a special case, when the output depends on the stream of Alice only.

2.4 Properties of the SC model

Several times in our proofs, we will consider an SC protocol and use it to solve problems in the standard models of CC. It is convenient to have a bound on how big the messages $m^{A/B}$ can be.

The length of the messages $m^{A/B}$ could be very big in the SC protocol, but we now show that the SC protocol can be simulated replacing them by length $S+1$ messages.

▶ **Proposition 5.** *In the SC model, we may always assume that the size of the messages is at most $S+1$ bits, up to redefining the transition functions $\Phi^{A/B}$.*

Proof. Consider a protocol Π with associated functions $\Phi^{A/B}, \Psi^{A/B}$.

The players can exchange their S size memory and the last input symbol instead of the actual messages. Hence, it is possible to redefine functions $\Phi^{A/B}, \Psi^{A/B}$ and directly assume messages have length $\leq S+1$. The new equations with $S+1$ bit messages would read $\sigma_{i+1}^{A/B} := \Phi^{A/B}(\Psi^{B/A}(\sigma_i^{B/A}, y_i, r), \sigma_i^{A/B}, r)$ and $m_{i+1}^{A/B} := \Psi^{A/B}(\sigma_{i+1}^{A/B}, x_{i+1}, r)$. ◀

Any protocol in the SC model can be simulated with another protocol in the usual CC model with a small overhead. Note that due to the implicit time in the SC model, we cannot immediately conclude that the SC model is harder than the usual communication model. Nevertheless, this time issue induces only an extra logarithmic factor.

▶ **Proposition 6.** *We can simulate any protocol Π in the SC model with parameters S, R, T with another protocol Π' in the normal communication model such that its communication cost $C(\Pi')$ is bounded as $C(\Pi') \leq T(1+2\log n)$ and $C(\Pi') \leq R(S+2\log n+1)$.*

We now compare the SC model to streaming algorithms, that is when there is a single player and a single stream. There are various ways to combine streams x and y in a single stream. Since x_i is presented to Alice at the same time as y_i to Bob, in a single player model x_i should be presented just before y_i to the player. This explains why we consider the interleaved streaming model.

▶ **Definition 7.** Let $\mathcal{S}^{\mathbf{int}}(f)$ be the amount of memory required for a streaming algorithm to compute f where the input stream is x, y interleaved, that is to say, $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$.

It turns out that interleaving streams instead of concatenating streams may affect the memory requirement of the function for a standard streaming algorithm by an exponential factor. A proof of the following result is provided in the full version of this paper.

▶ **Theorem 8.** *There is a function f such that $\mathcal{S}^{\mathbf{int}}(f) = \Omega(n)$, whereas there is a streaming algorithm to compute f with memory $O(\log n)$ when streams are concatenated.*

First let us observe that $S^{\mathbf{int}}(f)$ provides a lower bound in the SC model.

▶ **Proposition 9.** *Let f be a function. Then, any protocol in the SC model for the function f, where Alice and Bob use memories of size S, must have $2S \geq S^{\mathbf{int}}(f)$.*

It is natural to ask if there is a polynomial relation bounding the parameters S, R, T of a protocol in the SC model in terms of the streaming complexity $\mathcal{S}^{\mathbf{int}}(f)$ and the communication complexity $C(f)$, at least when $S = O(\mathcal{S}^{\mathbf{int}}(f))$. This appears to not hold in general. The following result is shown in the full version of the paper.

▶ **Theorem 10.** *There exists a function f such that $\mathcal{S}^{\mathbf{int}}(f) = C(f) = O(\log n)$ but any protocol computing f in the SC model must have $R \cdot S = \Omega(n)$. This holds for $S = \Omega(\log n)$.*

3 Communication primitives

We provide a general theorem that provides tight tradeoffs both in the deterministic and randomized case for a variety of functions, including $DISJ_n, IP_n, TRIBES_n$.

3.1 A general lower bound

In this section we show a general result that gives a lower bound for a large class of functions. We will obtain the lower bounds for the usual primitives $DISJ, IP, TRIBES$ as a corollary. We treat ε as a constant. The assumption we make is of a structural kind. Namely, as explained in Definition 12 we assume the function to be computed can be written in a depth-2 fashion, as a composition of an outer function G with inner gadgets g_ℓ.

▶ **Definition 11.** We call a function G on L variables *non trivial* if the following holds. There exists a word $a \in \{0,1\}^L$ such that for all ℓ there exists a postfix $b_\ell \in \{0,1\}^{L-\ell-1}$ such that $G(a_{\leq \ell} u b_\ell)$ depends on the bit u. More formally $G(a_{\leq \ell} 0 b_\ell) \neq G(a_{\leq \ell} 1 b_\ell)$.

This may look as a restrictive condition. In fact, most natural functions that depend on every bit are "non trivial" in this sense. For the function $\bigoplus$, a, b can be chosen arbitrarily. The functions OR and AND are also non trivial. For instance for AND, $a = b = 1^L$ will do.

We borrow the next definition from [4] (extending it slightly).

▶ **Definition 12** (Block-decomposable functions)**.** Let $I_1, \ldots, I_L$ be an interval partition of $[n]$, which we refer to as *blocks*. For $\ell \in [L]$, let $t_\ell = |I_\ell|$ be the length of I_ℓ. Given strings $x, y \in \{0,1\}^n$, write x^ℓ (resp. y^ℓ) for the restriction of x (resp. y) to indices in block I_ℓ. We say $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is *G-decomposable* with primitives (g_ℓ), where $G : \{0,1\}^L \to \{0,1\}$ and $g_\ell : \{0,1\}^{t_\ell} \times \{0,1\}^{t_\ell} \to \{0,1\}$, if for all inputs x, y we have $f(x,y) = G(g_1(x^1, y^1), g_2(x^2, y^2), \ldots, g_L(x^L, y^L))$.

Assuming f is G-decomposable, our goal is to show lower bound the communication needed to compute f in the SC model in terms of the communication complexity of each g_ℓ and the available memory.

▶ **Theorem 13.** *Assume the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is G-decomposable with primitives $(g_\ell)_{\ell \in L}$, and that G is non-trivial. Let $C_{\varepsilon+\delta}(g_\ell)$ be the worst-case randomized communication complexity of g_ℓ in the usual communication model. Then, any randomized protocol computing f with error ε in the SC model with S bits of memory, R expected communication rounds and T expected bits of total communication, must have*

$$R \geq \sum_{\ell \leq L} \frac{\delta C_{\varepsilon+\delta}(g_\ell) - S}{S + 2\log t_\ell + 1}, \quad T \geq \sum_{\ell \leq L} \frac{\delta C_{\varepsilon+\delta}(g_\ell) - S}{1 + 2\log t_\ell}.$$

We can get a similar bound in the deterministic case, where we use the deterministic communication complexity of the g_ℓ's. Last, if G is $\bigoplus$ we may remove δ from the above bounds, changing the complexities $C_{\varepsilon+\delta}(g_\ell)$ to $C^{avg}_{\varepsilon}(g_\ell)$.

3.2 Applications

Before proving Theorem 13, we give a few corollaries. Note that the upper bounds are trivial. Remind the function $TRIBES$, which is an AND of Set Intersections is defined by $TRIBES_n(x,y) := AND_{i \leq \sqrt{n}} \circ OR_{j \leq \sqrt{n}} (x_{ij} \bigwedge y_{ij})$.

▶ **Theorem 14.** *Any randomized protocol in the SC model that computes the function IP_n, $DISJ_n$, or $TRIBES_n$ and uses S memory and R communication rounds must have $R \cdot S = \Omega(n)$.*

Proof. We start with $DISJ_n$. We write $DISJ_n(x,y) = AND_{\ell=1}^{n/k}\left(DISJ_\ell(x^\ell, y^\ell)\right)$, and use the previous result with $G = AND$ and $g_\ell = DISJ_k$. It follows from Theorem 3 that $R_{\varepsilon+\delta}(DISJ_k) = \Omega(k)$. We omit the dependency on ε and δ in the term $\Omega()$, treating these parameters as fixed constants. The number of rounds for any randomized protocol in the SC model is at least $\sum_{\ell=1}^{\frac{n}{k}} \frac{\Omega(k)-S}{S+2\log k}$. We get the result choosing $k = \Omega(S)$.

In the case of IP_n, the function $f = IP_n$ is the composition of $G = \bigoplus$ over $\frac{n}{k}$ coordinates with $g_\ell = IP$ (for each $\ell \leq \frac{n}{k}$), over k coordinates. Theorem 3 gives $C_\varepsilon^{avg}(g_\ell) \geq \Omega(k)$. Theorem 13 yields the bound, taking $k = 10S$. We omit the case of $TRIBES_n$ as it follows from a similar argument. ◀

4 Approximate Matching in the Streaming Communication model

The main problem we consider is that of computing an approximate matching. The stream corresponds to edges (in an arbitrary order) of a bipartite graph $G = (P, Q, E)$ over vertex set P, Q, and the algorithm has to output a collection of edges which forms a matching. All edges in the output have to be in the original graph. In the vertex arrival setting, each vertex from Q arrives together with all the edges it belongs to. Our goal is to understand what is the best approximation ratio we can hope for, for a given memory (and message size).

We start by some notations. In a graph $G = (P, Q, E)$, if $U \subseteq P \cup Q$ and $V \subseteq P \cup Q$ are subsets of the vertices, we denote by $E(U, V) \subseteq E$ the edges with endpoints in U and V. We also denote by $OPT(G)$ the maximum size of a matching in G.

Observe now that when communication can occur at any step, the greedy algorithm, which is currently the best algorithm in the standard streaming model, can be implemented easily by having Alice communicate to Bob every time she adds an edge to her matching.

▶ **Proposition 15.** *The greedy algorithm, which achieves a 2-approximation, can be implemented using $n \log n$ bits of communication and n rounds in the SC model.*

Thus, we now focus on the one-way SC model, where the communication is restricted to happen once the streams have been fully read. In this setting, we will get different lower and upper bounds than in the streaming model. We start by a positive result.

▶ **Theorem 16** (Greedy matchings)**.** *If Alice sends Bob a maximal matching of her graph, then Bob can compute a 3-approximation. In particular a 3-approximation can be computed in a deterministic one-way SC protocol using $O(n \log n)$ memory and message size.*

Proof. Let G_1, G_2 be the respective graphs that Alice and Bob get, and let M_1, M_2 be their respective computed maximal matchings. We show that there is a matching in $M_1 \cup M_2$ of size at least $|OPT(G)|/3$.

The proof goes in two steps. Let ℓ be the size of $V(M_1 \cup M_2)$. We prove first that there is a matching of size $\geq \frac{\ell}{3}$ in the graph $M_1 \cup M_2$, and then that the number of edges in $OPT(G)$ is at most ℓ.

The first part is easy. Observe that $M_1 \cup M_2$ has maximal degree 2. Then there must be a matching of size at least $\ell/3$ from Theorem 7 in [6].

For the second part, we construct an injection $OPT(G) \hookrightarrow V(M_1 \cup M_2)$. Let $e = (u_1, u_2) \in OPT(G)$. Assume w.l.o.g. $e \in G_1$. Then either u_1 or u_2 is matched in M_1 (or both), by maximality of M_1. Map e to (one of) its matched endpoints in M_1. ◀

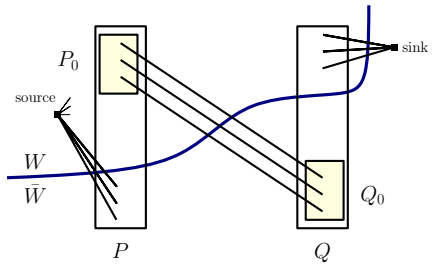


Figure 1 The figure shows how the maxflow mincut theorem is used to argue about the size of a matching in a bipartite graph G over vertex set $P \times Q$ drawn from a distribution μ. Only edges from the cut are drawn. The source and sink are added for the sake of the argument, they are not part of G.

Our lower bound is obtained using a black box reduction that we develop in the following sections. It is a direct consequence of the combination of Theorem 21 and Theorem 19.

▶ **Corollary 17** (A $(e+1)/(e-1)$ lower bound)**.** *Any protocol achieving a ratio of* $\frac{e+1}{e-1} - \eta \approx 2.16 - \eta$*, for some constant* η*, in the vertex arrival setting needs communication* $n^{1+\Omega(1/\log\log n)}$ *where the hidden constant in the* $\Omega(.)$ *depends on* η*.*

4.1 Hard distributions for streaming algorithms

Our notion of hard distribution is tailored to capture the distributions appearing in [17, 20]. They are distributions over streams of graphs, that is over graphs and edge orderings.

We will use the following definition for constructing families of hard distributions when $n \to \infty$ and α, η are fixed. Therefore $O()$ and $o()$ notations have to be understood in that context.

▶ **Definition 18** (Hard distribution)**.** A distribution μ over streams of bipartite graphs $G = (P, Q, E)$ is an $(\alpha, n, m(n), \eta)$-*hard distribution* when P and Q are sets of size n and the following holds

1. There is a cut $W, \overline{W}$ of vertices such that $|W \cap Q| + |\overline{W} \cap P| \leq (1 - \alpha + \eta)n$.
2. There is a matching M of size $(1-\eta)n$ in G that can be decomposed into $M_0 \cup M'$ such that
   - **(i)** $P_0 := V(M_0) \cap P$ and $Q_0 := V(M_0) \cap Q$ are of fixed size (in the support of μ) larger than $(\alpha - \eta)n$ and smaller than αn; and
   - **(ii)** $P_0 \subseteq W$ and $Q_0 \subseteq \overline{W}$.
3. Every streaming algorithm **Alg** with $o(m)$ bits of memory that outputs E^* with $E^* \subseteq E$ must satisfy $|E^* \cap E((W \cap P) \times (\overline{W} \cap Q))| = o(n)$ with probability $1 - o(1)$.

In particular, a streaming algorithm with small memory can only maintain on hard distributions a small fraction of edges $E((W \cap P) \times (\overline{W} \cap Q))$ and therefore of M_0. In addition, observe that for every matching E^* and cut $(W, \overline{W})$ (see Figure 1) it holds that $|M| \leq |W \cap Q| + |\overline{W} \cap P| + E((W \cap P) \times (\overline{W} \cap Q))$. Thus edges from $E(W \cap P \times \overline{W} \cap Q)$ are also crucial for obtaining a good matching.

The existence of hard distributions is ensured by [17, 20]. The hard distributions families are not exactly presented as we present them here. The following Theorem follows from results in [20] involving more parameters, for the purpose of the construction itself. We disregard those since we use the existence of hard distribution families as a black box.

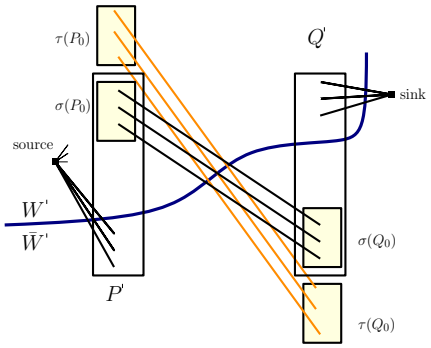


Figure 2 The construction of μ_2.

▶ **Theorem 19** ([20]). *For all $\eta > 0$ and n, there is a $(1/e, n, m(n), \eta)$-hard distribution μ over graphs of n vertices with $m(n) = n^{1+\Omega(\frac{1}{\log\log n})}$, where the notation $\Omega(.)$ hides a dependency on η.*

Sketch of Proof. Specifically, point (1) of our definition follows from [20, Lemma 13], point (2) follows from [20, Claim 12], and point (3) follows from [20, Lemma 14]. ◀

4.2 Lifting hard distributions to streaming communication protocols

Let μ be an (α, n, m, η)-hard distribution for streaming algorithms. We will show how to extend it to a distribution μ_2 for the two party version of the approximate matching problem. At a high level, we give the players two copies of the same graph G randomly chosen according to μ, but embedded into two different but overlapping subsets of vertices (see Figure 2). The non-overlapping parts correspond to edges from $E(W \cap P \times \overline{W} \cap Q)$ (see Definition 18), and are therefore hard to maintain, but necessary to setup a large matching.

From now on, identify P with the set $[n]$. Set $\beta := 1 + \alpha$. Our labelings are defined over vertex set $P' \times Q' := [\beta n] \times [\beta n]$, and are encoded by injections from $[n]$ to $[\beta n]$ (where for simplicity βn is understood as an integer).

Given a hard distribution μ, we define a distribution μ_2 as follows.

▶ **Definition 20** (The distribution μ_2). Let μ be a hard distribution, where P and Q are identified with $[n]$. Then sampling a bipartite graph over vertex set $P' \times Q' = [\beta n] \times [\beta n]$ from μ_2 is defined as follows

- Sample $G \sim \mu$. Let $(W, \overline{W})$ and P_0, Q_0 be the corresponding cut and sets from Definition 18.
- Sample σ, τ uniformly at random such that $\sigma(P_0) \cup \tau(P_0) = \emptyset = \sigma(Q_0) \cup \tau(Q_0)$ are disjoint, and σ, τ are equal on $P \setminus P_0$. Such injections σ, τ are called *G-compatible*.

In addition, define $G_\sigma := (\sigma(P), \sigma(Q), E_\sigma)$, where $E_\sigma = \{(\sigma(u), \sigma(v)) \mid (u, v) \in E\}$, and G_τ similarly. Alice is given G_σ and Bob is given G_τ with the same order as under μ.

In this construction, observe that edges sent to Alice and Bob may overlap. In fact the distribution can be tweaked to make edges disjoint using a simple gadget, while preserving the same lower bound (see the full version of this paper). We can now state our main result for the reduction.

▶ **Theorem 21** (Generic reduction). *If there exists an (α, n, m, η)-hard distribution for approximate matching, then any protocol in the one-way SC model whose approximation ratio is $\frac{1-\alpha}{1+\alpha} - O(\eta)$ has to use $\Omega(m)$ memory.*

Proof. Define a cut for the two player instance as $W' := \sigma(W) \cup \tau(P_0)$, $\overline{W}' := \sigma(\overline{W}) \cup \tau(Q_0)$. It follows from the max-flow min-cut argument that any matching E^* has size at most $|E^*| \leq |\overline{W}' \cap Q'| + |W' \cap P'| + |E^* \cap E(W' \cap P' \times \overline{W}' \cap Q')|$.

Moreover note that by construction $|\overline{W}' \cap Q'| = |\sigma(\overline{W}) \cap Q'|$. Indeed $\tau(P_0) \cap Q' = \emptyset$. Then we can write $|\sigma(\overline{W}) \cap Q'| = |W \cap Q|$ and similarly for $|W' \cap P'|$ (see also Figure 2). It follows that

$$|\overline{W}' \cap Q'| + |\overline{W}' \cap P'| = |\overline{W} \cap Q| + |\overline{W} \cap P| \leq (1 - \alpha + \eta)\, n.$$

The set of edges the protocol outputs is included in $E^*_A \cup E^*_B$ by definition. Under the high probability event that these sets only have an overlap of $o(n)$ with the "important edges" $E(W' \cap P \times \overline{W}' \cap Q)$ (see Lemma 22 below) then, if E^* is the output matching by a protocol using $o(m)$ memory, then using Lemma 22 the matching $E^* \subseteq E^*_A \cup E^*_B$ is of size at most $(1 - \alpha + \eta)\, n + o(n) \leq (1 - \alpha + 2\eta)\, n$ (for large enough n).

On the other hand, there is a matching between $\sigma(P)$ and $\sigma(Q)$ of size $(1 - \eta)n$ and a matching of size $(\alpha - \eta)n$ between $\tau(P_0)$ and $\tau(Q_0)$ (using Point (2) in the definition of a hard distribution for approximate matching, Definition 18). We identified $\sigma(P) \sqcup \tau(P_0)$ with $[\beta n]$ and hence under μ_2 there is a matching of size $(\alpha - \eta)n + (1 - \eta)n = \beta n - 2\eta n$. This shows that the approximation ratio of a protocol using $o(m)$ memory is smaller than $\frac{\beta n - 2\eta n}{(1 - \alpha + 2\eta)} = \frac{1+\alpha}{1-\alpha} - O(\eta)$. ◀

▶ **Lemma 22.** *Let* $\boldsymbol{Alg}$ *be a protocol for the two party case, i.e. a pair of algorithms for Alice and Bob* $\boldsymbol{Alg} = (\boldsymbol{Alg}_A, \boldsymbol{Alg}_B)$*. Let* E^*_A *(resp.* E^*_B*) denote the edges* $\boldsymbol{Alg}_A$ *(resp.* $\boldsymbol{Alg}_B$*) outputs, assuming only* $o(m)$ *memory is used. With probability* $1 - o(1)$ *over the choice of* σ, τ, G*, or alternatively with probability* $1 - o(1)$ *under* μ_2*, it holds that*

$$|E^*_A \cap E(W' \cap P' \times \overline{W}' \cap Q')| = o(n), \quad \textit{and similarly} \quad |E^*_B \cap E(W' \cap P' \times \overline{W}' \cap Q')| = o(n).$$

Proof. The proof consists in building from $\mathbf{Alg}_A$, and similarly $\mathbf{Alg}_B$, a streaming algorithm for μ. Indeed, for inputs over vertices $[n]$ distributed according to μ, simply pick a random σ apply it to the input, and run $\mathbf{Alg}_A$ on the graph G_σ. Then output $E^*_0 := \sigma^{-1}(E^*_A)$.

First observe that the distribution of σ and the distribution of G are independent. Therefore, conditioned on G, the injection σ is uniform. It follows that μ_2's first marginal is also the distribution of G_σ, where $G \sim \mu$ and σ is uniform and independent.

Then, using Definition 18, with probability $1 - o(1)$ over the choice of σ and $G \sim \mu$, we obtain that $|E^*_A \cap E(W' \cap P' \times \overline{W}' \cap Q')| = |E^*_0 \cap E(W \cap P \times \overline{W} \cap Q)| = o(n)$, which concludes the proof. ◀

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

2 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms*, pages 1345–1364, 2016.

3 László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Proceedings of the 27th IEEE Foundations of Computer Science*, pages 337–347, 1986.

4 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

5 Paul Beame, Martin Tompa, and Peiyuan Yan. Communication-space tradeoffs for unrestricted protocols. In *Proceedings of 31st Foundations of Computer Science*, pages 420–428, 1990.

6 Therese C. Biedl, Erik D. Demaine, Christian A. Duncan, Rudolf Fleischer, and Stephen G. Kobourov. Tight bounds on maximal and maximum matchings. *Discrete Mathematics*, 285(1-3):7–15, 2004.

7 Joshua E. Brody, Shiteng Chen, Periklis A. Papakonstantinou, Hao Song, and Xiaoming Sun. Space-bounded communication complexity. In *Proceedings of the 4th Innovations in Theoretical Computer Science*, pages 159–172, 2013.

8 Ho-Leung Chan, Tak-Wah Lam, Lap-Kei Lee, and Hing-Fung Ting. Continuous monitoring of distributed data streams over a time-based sliding window. *Algorithmica*, 62(3):1088–1111, 2011.

9 Graham Cormode and Minos N. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *International Conference on Very Large Data Bases*, pages 13–24, 2005.

10 Graham Cormode, Minos N. Garofalakis, S. Muthukrishnan, and Rajeev Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Special Interest Group on Management of Data*, pages 25–36, 2005.

11 Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 1076–1085, 2008.

12 Graham Cormode, S. Muthukrishnan, Ke Yi, and Qin Zhang. Continuous sampling from distributed streams. *J. ACM*, 59(2):10:1–10:25, 2012.

13 Graham Cormode, S. Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Proceedings of the 22nd International Conference on Data Engineering*, page 57, 2006.

14 Jon Feldman, S. Muthukrishnan, Anastasios Sidiropoulos, Clifford Stein, and Zoya Svitkina. On the complexity of processing massive, unordered, distributed data. *CoRR*, abs/cs/0611108, 2006.

15 Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures*, pages 281–291, 2001.

16 Phillip B. Gibbons and Srikanta Tirthapura. Distributed streams algorithms for sliding windows. In *Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures*, pages 63–72, 2002.

17 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the 23d ACM-SIAM Symposium on Discrete Algorithms*, pages 468–485, 2012.

18 Zengfeng Huang, Ke Yi, and Qin Zhang. Randomized algorithms for tracking distributed count, frequencies, and ranks. In *Proceedings of the 31st Symposium on Principles of Database Systems*, pages 295–306, 2012.

19 Russell Impagliazzo and Ryan Williams. Communication complexity with synchronized clocks. In *Proceedings of the 25th IEEE Conference on Computational Complexity*, pages 259–269, 2010.

20 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms*, pages 1679–1697, 2013.

21 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751, 2014. doi:10.1137/1.9781611973402.55.

22 Christian Konrad. Maximum matching in turnstile streams. In *Proceedings of 23rd European Symposium on Algorithms*, pages 840–852, 2015.

23 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

24 Tak Wah Lam, Prasoon Tiwari, and Martin Tompa. Trade-offs between communication and space. *Journal of Computer and System Sciences*, 45(3):296–315, 1992.

25 Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the 46th ACM Symposium on Theory of Computing*, pages 174–183, 2014.

26 Zhenming Liu, Bozidar Radunovic, and Milan Vojnovic. Continuous distributed counting for non-monotonic streams. In *Proceedings of the 31st ACM Symposium on Principles of Database Systems*, pages 307–318, 2012.

27 S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 2(1):117–236, 2005.

28 David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th ACM Symposium on Theory of Computing*, pages 941–960, 2012.

29 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th ACM Symposium on Theory of Computing*, pages 209–213, 1979.

Testable Bounded Degree Graph Properties Are Random Order Streamable

Morteza Monemizadeh[*1], S. Muthukrishnan[2], Pan Peng[†3], and Christian Sohler[‡4]

1 **Department of Computer Science, Goethe-Universität Frankfurt, Frankfurt, Germany**
monemi@ae.cs.uni-frankfurt.de
2 **Rutgers University, Piscataway, NJ, USA**
muthu@cs.rutgers.edu
3 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
pan.peng@univie.ac.at
4 **Department of Computer Science, TU Dortmund, Dortmund, Germany**
christian.sohler@tu-dortmund.de

Abstract

We study which property testing and sublinear time algorithms can be transformed into graph streaming algorithms for random order streams. Our main result is that for bounded degree graphs, any property that is constant-query testable in the adjacency list model can be tested with *constant space* in a single-pass in random order streams. Our result is obtained by estimating the distribution of local neighborhoods of the vertices on a random order graph stream using constant space.

We then show that our approach can also be applied to constant time approximation algorithms for bounded degree graphs in the adjacency list model: As an example, we obtain a constant-space single-pass random order streaming algorithms for approximating the size of a maximum matching with additive error ϵn (n is the number of nodes).

Our result establishes for the first time that a large class of sublinear algorithms can be simulated in random order streams, while $\Omega(n)$ space is needed for many graph streaming problems for adversarial orders.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Graph streaming algorithms, graph property testing, constant-time approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.131

1 Introduction

Very large and complex networks abound. Some of the prominent examples are gene regulatory networks, health/disease networks, and online social networks like Facebook, Google+, Linkedin and Twitter. The interconnectivity of neurons in human brain, relations in database systems, and chip designs are some further examples. Some of these networks can be quite large and it may be hard to store them completely in the main memory and some

* Partially supported by DFG grants ME 2088/3-(1/2) and ME 2088/4-1.
† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.
‡ Supported by ERC Starting Grant 307696.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 131; pp. 131:1–131:14

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

may be too large to be stored at all. However, these networks contain valuable information that we want to reveal. For example, social networks can provide insights into the structure of our society, and the structure in gene regulatory networks might yield insights into diseases. Thus, we need algorithms that can analyze the structure of these networks quickly.

One way to approach this problem is to design graph streaming algorithms [16, 1]. A graph streaming algorithm gets access to a stream of edges in some order and exactly or approximately solves problems on the graph defined by the stream. The challenge is that a graph streaming algorithm should use space sublinear in the size of the graph. We focus on algorithms that make only *one pass* over the graph stream. It has been shown that many natural graph problems require $\Omega(n)$ space in the *adversarial order* model where n is the number of nodes in the graph and the edges can arrive in arbitrary order (see eg.,[8, 9]), and thus most of previous work has focused on the *semi-streaming* model, in which the algorithms are allowed to use $O(n \cdot \operatorname{poly}\log n)$ space. However, in many interesting applications, the graphs are sparse and so they can be fully stored in the semi-streaming model making this model useless in this setting. This raises the question *whether there are at least some natural conditions under which one can solve graph problems with space $o(n)$, possibly even $\log^{O(1)} n$ or constant.*

One such condition that recently received increasing attention is that the edges arrive in *random order*, i.e. in the order of a uniformly random permutation of the edges (e.g., [5, 22, 19]). Uniformly random or near-uniformly random ordering is a natural assumption and can arise in many contexts. Indeed, previous work has shown that some problems that are hard for adversarial streams can be solved in the random order model. Konrad et al. [22] gave single-pass semi-streaming algorithms for maximum matching for bipartite and general graphs with approximation ratio strictly larger than $1/2$ in the random order semi-streaming model, while no such approximation algorithm is known in the adversary order model. Kapralov et al. [19] gave a polylogarithmic approximation algorithm in polylogarithmic space for estimating the size of maximum matching of an unweighted graph in one pass over a random order stream. Assadi et al. [2] recently showed that in the adversarial order and *dynamic* model where edges can be both inserted and deleted, any polylogarithmic approximation algorithm of maximum matching size requires $\tilde{\Omega}(n)$ space. On the other hand, Chakrabarti et al. [5] presented an $\Omega(n)$ space lower bound for any single pass algorithm for graph connectivity in the random order streaming model, which is very close to the optimal $\Omega(n \log n)$ space lower bound in the adversarial order model [30]. In general, it is unclear which graph problems can be solved in random order streams using much smaller space than what is required for adversarially ordered streams.

An independent area of research is *property testing*, where with certain *query* access to an object (eg., random vertices or neighbors of a vertex for graphs), there are algorithms that can determine if the object satisfies a certain property, or is far from having such a property [29, 11, 12]. The area of property testing has seen fundamental results, including testing various general graph properties. For example, it has been shown that many interesting properties (including connectivity, planarity, minor-freeness, hyperfiniteness) of bounded degree graphs can be tested with a constant number of queries [12, 3, 25]. Another very related area of research is called *constant-time* (or in general, *sublinear-time*) *approximation* algorithms, where we are given query access to an object (for example a graph) and the goal is to approximate the objective value of an optimal solution. For example, in bounded degree graphs, one can approximate the cost of the optimal solution with constant query complexity for some fundamental optimization problems (e.g., minimum spanning tree weight [6], maximal matching size [26]; see also Section 1.3).

A fundamental question is if such results from property testing and constant-time approximation algorithms will lead to better graph streaming algorithms. Huang and Peng [17] recently considered the problem of estimating the minimum spanning tree weight and property testing for general graphs in dynamic and adversarial order model. They showed that a number of properties (e.g., connectivity, cycle-freeness) of general n-vertex graphs can be tested with space complexity $O(n^{1-\varepsilon})$ and one can $(1+\varepsilon)$-approximate the weight of minimum spanning tree with similar space guarantee. Furthermore, there exist $\Omega(n^{1-O(\varepsilon)})$ space lower bounds for these problems that hold even in the insertion-only model [17].

1.1 Overview of Results

In this paper we provide a general framework that transforms bounded-degree graph property testing to very space-efficient random order streaming algorithms.

To formally state our main result, we first review some basic definitions of graph property testing. A *graph property* is a property that is invariant under graph isomorphism. Let $G = (V, E)$ be a graph with maximum degree upper bounded by a constant d, and we also call G a *d-bounded* graph. In the *adjacency list model* for (bounded-degree) graph property testing, we are given query access to the adjacency list of the input d-bounded graph $G = (V, E)$. That is, for any vertex $v \in V$ and index $i \leq d$, one can query the ith neighbor (if exists) of vertex v in constant time. Given a property Π, we are interested in testing if a graph G satisfies Π or is *ε-far* from satisfying Π while making as few queries as possible, where G is said to be ε-far from satisfying Π if one has to insert/delete more than εdn edges to make it satisfy Π. We call a property *constant-query testable* if there exists a testing algorithm (also called *tester*) for this property such that the number of performed queries depends only on parameters ε, d and is independent of the size of the input graph.

Given a graph property Π, we are interested in *approximately* testing it in a single-pass stream with a goal similar to the above. That is, the algorithm uses little space and with high constant probability, it accepts the input graph G if it satisfies P and rejects G if it is ε-far from satisfying P (see Section 4 for formal definitions). Our main result is as follows.

▶ **Theorem 1.** *Any d-bounded graph property that is constant-query testable in the adjacency list model can be tested in the uniformly random order streaming model with constant space.*

To the best of our knowledge, this is the first non-trivial graph streaming algorithm with constant space complexity (measured in the number of *words*, where a word is a space unit large enough to encode an ID of any vertex in the graph.) By the constructions in [17], there exist graph properties (e.g., connectivity and cycle-freeness) of d-bounded graphs such that any single-pass streaming algorithm in the insertion-only and *adversary order* model must use $\Omega(n^{1-O(\varepsilon)})$ space. In contrast to this lower bound, our main result implies that d-bounded connectivity and cycle-freeness can be tested in constant space in the random order stream model, since they are constant-query testable in the adjacency list model [12].

Our approach also works for simulating *constant-time approximation* algorithms as graph streaming algorithms with constant space. For a minimization (resp., maximization) optimization problem P and an instance I, we let $\mathrm{OPT}(I)$ denote the value of some optimal solution of I. We call a value x an (α, β)-approximation for the problem P, if for any instance I, it holds that $\mathrm{OPT}(I) \leq x \leq \alpha \cdot \mathrm{OPT}(I) + \beta$ (resp., $\frac{\mathrm{OPT}(I)}{\alpha} - \beta \leq x \leq \mathrm{OPT}(I)$). For example, it is known that there exists a constant-query algorithm for $(1, \varepsilon n)$-approximating the maximal matching size of any n-vertex d-bounded graph [26]. That is, the number of queries made by the algorithm is independent of n and only depends on ε, d. As an application, we show:

▶ **Theorem 2.** *Let $0 < \varepsilon < 1$ and d be constants. Then there exists an algorithm that uses constant space in the random order model, and with probability $2/3$, $(1, \varepsilon n)$-approximates the size of some maximal matching in d-bounded graphs.*

We also remark that in a similar way, many other sublinear time algorithms for bounded degree graphs can be simulated in random order streams. Finally, our results can actually be extended to a model which requires weaker assumptions on the randomness of the order of edges in the stream, but we describe our results for the uniformly random order model, and leave the remaining details for later.

1.2 Technical Overview

The local neighborhood of depth k of a vertex v is the subgraph rooted at v and induced by all vertices of distance at most k from v. We call such a rooted subgraph a *k-disc*. Suppose that we are given a sufficiently large graph G whose maximum degree d is constant. This means that for any constant k, a k-disc centered at an arbitrary vertex v in G has constant size. Now assume that there exists an algorithm $\mathcal{A}$ that, independent of the labeling of the vertices of G, accesses G by querying random vertices and exploring their k-discs. We observe that any constant-query property tester (see for example [13, 7]) falls within the framework of such an algorithm. If instead of the graph G we are given the distribution of k-discs of the vertices of G, we can use this distribution to simulate the algorithm $\mathcal{A}$ and output with high probability the same result as executing the algorithm $\mathcal{A}$ on G itself. Thus, the problem of developing constant-query property testers in random order streams can be reduced to the problem of designing streaming algorithms that approximate the distribution of k-discs in G.

The main technical contribution of this paper is an algorithm that given a random order stream S of edges of an underlying d-bounded degree graph G, approximates the distribution of k-discs of G up to an additive error of δ. We would like to mention that if the edges arrive in adversarial order, any algorithm that approximates the distribution of k-discs of G requires almost linear space [32, 17], hence the assumption of random order streams (or something similar) is necessary to obtain our result.

Now in order to approximate the distribution of k-discs of the graph G we do the following. We proceed by sampling vertices uniformly at random and then perform a BFS for each sampled vertex using the arrival of edges along the stream S. Note that the new edges of the stream S that do not connect to the currently explored vertices are discarded. Let us call the k-disc that is observed by doing such a BFS from some vertex v to be Δ_1. Due to possibility of missing edges during the BFS, this subgraph may be different from the true k-disc Δ_2 rooted at v.

Fortunately, since the edges arrive in a uniformly random order, we can infer the conditional probability $\Pr[\Delta_1|\Delta_2]$. That is, given the true rooted subgraph Δ_2, we can compute the conditional probability of seeing a rooted subgraph Δ_1 in a random order stream when the true k-disc is Δ_2.

We define the partial order on the set of k-discs given by $\Delta_1 \preccurlyeq \Delta_2$ whenever Δ_1 is a root-preserving isomorphic subgraph of Δ_2. For every two k-discs Δ_1 and Δ_2 with $\Delta_1 \preccurlyeq \Delta_2$ we compute the conditional probability $\Pr[\Delta_1|\Delta_2]$. Using the set of all conditional probabilities $\Pr[\Delta_1|\Delta_2]$ we can estimate or approximate the distribution of k-discs of the graph G whose edges are revealed according to the stream S. In order to simplify the analysis of our algorithm, we require a natural independence condition for non-intersecting k-discs. Finally, we use the approximated distribution of k-discs to simulate the algorithm $\mathcal{A}$ by the machinary that we explained above.

We remark that the idea of using a partial order to compute a distribution of k-discs in bounded degree graphs has first been used in [7]. However, the setting in [7] was quite different as it dealt with directed graphs where an edge can only be seen from one side (and the sample sizes required in that paper were only slightly sublinear in n).

1.3 Other Related Work

Feigenbaum et al. [10] initiated the study of property testing in streaming model, and they gave efficient testers for some properties of a sequence of data items (rather than graphs as we consider here). Bury and Schwiegelshohn [4] gave a lower bound of $n^{1-O(\varepsilon)}$ on the space complexity of any algorithm that $(1-\varepsilon)$-approximates the size of maximum matching in adversarial streams. Kapralov et al. [20] showed that in random streams, $\tilde{\Omega}(\sqrt{n})$ space is necessary to distinguish if a graph is bipartite or 1/2-far from being bipartite. Previous work has extensively studied streaming graph algorithms in both the insertion-only and dynamic models, see the recent survey [24].

In the framework of d-bounded graph property testing, it is now known that many interesting properties are constant-query testable in the adjacency list model, including k-edge connectivity, cycle-freeness, subgraph-freeness [12], k-vertex connectivity [33], minor-freeness [15, 3], matroids related properties [18, 31], hyperfinite properties [25], subdivision-freeness [21]. Constant-time approximation algorithms in d-bounded graphs are known to exist for a number of fundamental optimization problems, including $(1+\varepsilon)$-approximating the weight of minimum spanning tree [6], $(1,\varepsilon n)$-approximating the size of maximal/maximum matching [26, 34], $(2,\varepsilon n)$-approximating the minimum vertex cover size [28, 23, 27], $(O(\log d),\varepsilon n)$-approximating the minimum dominating set size [28, 26]. For d-bounded minor-free graphs, there are constant-time $(1,\varepsilon n)$-approximation algorithms for the size of minimum vertex cover, minimum dominating set and maximum independent set [15].

2 Preliminaries

Let $G=(V,E)$ be an n-vertex graph with maximum degree upper bounded by some constant d, where we often identify V as $[n] := \{1,\cdots,n\}$. We also call such a graph d-bounded graph. In this paper, we will assume the algorithms have the knowledge of n, d. We assume that G is represented as a sequence of edges, which we denote as $\textsc{Stream}(G)$.

Graph k-discs. Let $k \geq 1$. The k-disc around a vertex v is the subgraph rooted at vertex v and induced by the vertices within distance at most k from v. Note that for an n-vertex graph, there are exactly n k-discs. Let $\mathcal{H}_{d,k} = \{\Delta_1,\cdots,\Delta_N\}$ be the set of all k-disc isomorphism types, where $N = N_{d,k}$ is the number of all such types (and is thus a constant). In the following, we will refer to a k-disc of some vertex v in the graph G as $\mathrm{disc}_{k,G}(v)$ and a k-disc type as Δ. Note that for every vertex v, there exists a unique k-disc type $\Delta \in \mathcal{H}_{d,k}$ such that $\mathrm{disc}_{k,G}(v)$ is isomorphic to Δ, denoted as $\mathrm{disc}_{k,G}(v) \cong \Delta$. (Throughout the paper, we call two rooted graphs H_1, H_2 isomorphic to each other if there is a root-preserving mapping from the vertex set of H_1 to the vertex set of H_2.)

We further assume that all the elements in $\mathcal{H}_{d,k}$ are ordered according to the natural partial order among k-disc types. More specifically, for any two k-disc types Δ_i, Δ_j, we let $\Delta_i \succcurlyeq \Delta_j$ (or equivalently, $\Delta_j \preccurlyeq \Delta_i$) denote that Δ_j is root-preserving isomorphic to some subgraph of Δ_i. Then we order all the k-disc types $\Delta_1,\cdots,\Delta_N$ such that if $\Delta_i \succcurlyeq \Delta_j$, then $i \leq j$. Let $\mathcal{G}(j)$ denote all the indices i, except j itself, such that $\Delta_i \succcurlyeq \Delta_j$.

Locally random order streams. Let Σ_E denote the set of all permutations (or orderings) over the edge set E. Note that each $\sigma \in \Sigma_E$ determines the order of edges arriving from the stream. Let $\mathcal{D} = \mathcal{D}(\Sigma_E)$ denote a probability distribution over Σ_E. In particular, we let $\mathcal{U} = \mathcal{U}(\Sigma_E)$ denote the uniform distribution over Σ_E. Given a stream σ of edges, we define the *observed k-disc of v from the stream*, denoted as $\mathrm{disc}_k(v, \sigma)$, to be the subgraph rooted at v and induced by all edges that are sequentially collected from the stream and the endpoints of which are within distance at most k to v. This is formally defined in the following algorithm STREAM_k-DISC.

Algorithm 1 The observed k-disc of v from the stream

```
1: procedure STREAM_k-DISC(STREAM(G),k,v)
2:     U ← {v}, ℓ_v = 0, F ← ∅
3:     for (u, w) ← next edge in the stream do
4:         if exactly one of u, w, say u, is contained in U then
5:             if ℓ_u ≤ k − 1 then
6:                 U ← U ∪ {w}, F ← F ∪ {(u, w)}
7:                 for x ∈ U do
8:                     ℓ_x ← the distance between x and v in the graph G' = (U, F)
9:                 end for
10:            end if
11:        else if both u, v are contained in U then
12:            F ← F ∪ {(u, w)}
13:            for x ∈ U do
14:                ℓ_x ← the distance between x and v in the graph G' = (U, F)
15:            end for
16:        end if
17:    end for
18:    return disc_k(v, σ) ← the subgraph rooted at v and induced by all edges in F
19: end procedure
```

Now we formally define a locally random distribution on the order of edges.

▶ **Definition 3.** Let $d, k > 0$. Let $G = (V, E)$ be a d-bounded graph. Let $\mathcal{D}$ be a distribution over all the orderings of edges in E. Let $\Lambda_k = \{\lambda(\Delta_i|\Delta_j) : 0 \le \lambda(\Delta_i|\Delta_j) \le 1, \Delta_j \succcurlyeq \Delta_i, 1 \le i, j \le N\}$ be a set of real numbers in $[0, 1]$. We call $\mathcal{D}$ a *locally random Λ_k-distribution* over G with respect to k-disc types, if for σ sampled from $\mathcal{D}$, the following conditions are satisfied:

1. (*Conditional probabilities*) For any vertex v with k-disc isomorphic to Δ_j, the probability that its observed k-disc $\mathrm{disc}_k(v, \sigma) \cong \Delta_i$ is $\lambda(\Delta_i|\Delta_j)$, for any i such that $\Delta_j \succcurlyeq \Delta_i$.
2. (*Independence of disjoint k-discs*) For any two disjoint k-discs $\mathrm{disc}_{k,G}(v)$ and $\mathrm{disc}_{k,G}(u)$, their observed k-discs $\mathrm{disc}_k(v, \sigma)$ and $\mathrm{disc}_k(u, \sigma)$ are independent.

Note that the set Λ_k cannot be an arbitrary set, as there might be no distribution satisfying the above condition. On the other hand, if there indeed exists a distribution satisfying the condition with numbers in Λ_k, then we call the set Λ_k *realizable*. In the following, we call a stream a *locally random order stream* if there exists a family of realizable sets $\Lambda = \{\Lambda_k\}_{k\ge1}$, such that the edge order is sampled from some locally random Λ_k-distribution with respect to k-disc types, for any integer $k \ge 1$. We have the following lemma.

▶ **Lemma 4.** *Let $d \geq 1$. For any $k \geq 1$, there exists $n_0 = n_0(k,d)$, such that for $n \geq n_0$, any d-bounded n-vertex graph $G = (V,E)$, the uniform permutation $\mathcal{U}$ over E is a locally random Λ_k-distribution over G with respect to k-disc types, for some realizable $\Lambda_k := \{\lambda(\Delta_i|\Delta_j) : 0 \leq \lambda(\Delta_i|\Delta_j) \leq 1, \Delta_j \succcurlyeq \Delta_i, 1 \leq i,j \leq N\}$. Furthermore, if we let $\kappa := \max_{i,j:\Delta_j \succcurlyeq \Delta_i} \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)}$, $\lambda_{\min} := \min_{i \leq N} \lambda(\Delta_i|\Delta_i)$, then $\kappa \leq 2^{2d^{k+1}}$, $\lambda_{\min} \geq \frac{1}{(2d^{k+1})!}$.*

Proof. Note that for any vertex v with $\mathrm{disc}_{k,G}(v) \cong \Delta_j$, the probability that the observed k-disc of v is isomorphic to Δ_i is exactly the fraction of orderings σ such that $\mathrm{disc}_k(v,\sigma) \cong \Delta_i$, where $\Delta_j \succcurlyeq \Delta_i$. We use such a fraction, which is a fixed real number, to define $\lambda(\Delta_i|\Delta_j)$. Observe that for an ordering σ sampled from $\mathcal{U}$, it directly satisfies the second condition Item 2 in Definition 3. Since there are at most $2d^{k+1}$ edges in any k-disc, the probability of observing a full k-disc is at least $\frac{1}{(2d^{k+1})!}$, that is, $\lambda_{\min} \geq \frac{1}{(2d^{k+1})!}$. Furthermore, since the k-disc type Δ_j might contain at most $\binom{|E(\Delta_j)|}{|E(\Delta_i)|} \leq 2^{2d^{k+1}}$ different subgraphs that are isomorphic to Δ_i, it holds that $\lambda(\Delta_i|\Delta_j) \leq \sum_{F:F \text{ subgraph of } \Delta_j, F \cong \Delta_i} \lambda(\Delta_i|\Delta_i) \leq 2^{2d^{k+1}} \lambda(\Delta_i|\Delta_i)$ for any i,j such that $\Delta_j \succcurlyeq \Delta_i$. This completes the proof of the lemma. ◀

The above lemma shows that the uniformly random order stream is a special case of a locally random order stream. Another natural class of locally random order stream is ℓ-wise independent permutation of edges for any $\ell = \omega_n(1)$ (i.e., any function that tends to infinity as n goes to infinity) for n-vertex bounded degree graphs, but for our qualitative purposes here, it suffices to consider uniformly random order streams.

3 Approximating the k-Disc Type Distribution

In this section, we show how to approximate the distribution of k-disc types of any d-bounded graph in locally random order streams.

Recall that for any k,d, we let $N = N_{d,k}$ be the constant denoting the number of all possible k-disc isomorphism types. For any $i \leq N$, let V_i be the set of vertices from V with k-disc isomorphic to Δ_i in the input graph G, that is, $V_i := \{v | v \in V, \mathrm{disc}_{k,G}(v) \cong \Delta_i\}$. Note that $f_i = \frac{|V_i|}{n}$ is the fraction of vertices with k-disc isomorphic to Δ_i.

▶ **Lemma 5.** *Let $G = (V,E)$ be a d-bounded graph presented in a locally random order stream defined by a Λ_k-distribution $\mathcal{D}$ over G with respect to k-disc types, for some integer k. Let $\kappa := \max_{i,j:\Delta_j \succcurlyeq \Delta_i} \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)}$, $\lambda_{\min} := \min_{i \leq N} \lambda(\Delta_i|\Delta_i)$. Then for any constant $\delta > 0$, there exists an algorithm that uses $O(\frac{\kappa^{2N} \cdot d^{3k+2} \cdot 3^{3N+1}}{\delta^2 \lambda_{\min}})$ space, and with probability $\frac{2}{3}$, for any $i \leq N$, approximates the fraction f_i of vertices with k-disc isomorphic to Δ_i in G with additive error δ.*

Proof. Our algorithm is as follows. We first sample a constant number of vertices, which are called centers. Then for each center v, we collect the observed k-disc of v from the stream. Then we postprocess all the collected edges and use the corresponding empirical distribution of k-disc types of all centers to estimate the distribution of k-disc types of the input graph. The formal description is given in Algorithm 2.

Note that since there are $s = \frac{8\kappa^{2N} \cdot d^{2k+1} \cdot 3^{3N+1}}{\delta^2 \lambda_{\min}}$ vertices in A and only edges that belong to the k-discs of these vertices will be collected by our algorithm, the space complexity of the algorithm is $O(sd^{k+1}) = O(\frac{\kappa^{2N} \cdot d^{3k+2} \cdot 3^{3N+1}}{\delta^2 \lambda_{\min}})$, which is constant.

Now we show the correctness of the algorithm.

Algorithm 2 Approximating the distribution of k-disc types

procedure k-DISC_DISTRIBUTION(STREAM(G),Λ_k,n, d, k, δ)
  sample a set A of $s := \frac{8\kappa^{2N} \cdot d^{2k+1} \cdot 3^{3N+1}}{\delta^2 \lambda_{\min}}$ vertices uniformly at random
  for each $v \in A$ **do**
    $H_v \leftarrow$ STREAM_k-DISC(STREAM(G),v,k) ▷ to collect observed k-disc of v
  end for
end procedure

procedure POSTPROCESSING
  $H \leftarrow$ the graph spanned by $\cup_{v \in A} H_v$
  for $i = 1$ to N **do**
    $Y_i \leftarrow |\{v : v \in A, \mathrm{disc}_{k,H}(v) \cong \Delta_i\}|/s$
    $X_i \leftarrow (Y_i - \sum_{j \in \mathcal{G}(i)} X_j \cdot \lambda(\Delta_i|\Delta_j)) \cdot \lambda^{-1}(\Delta_i|\Delta_i)$.
  end for
  return $X_1, \cdots, X_N$
end procedure

We let $A \sim \mathcal{U}_V$ denote that A is the set of s vertices sampled uniformly at random from V. For any $i \leq N$, let A_i be the set of vertices from A with k-disc isomorphic to Δ_i in the input graph G, that is, $A_i := \{v | v \in A, \mathrm{disc}_{k,G}(v) \cong \Delta_i\}$. Note that $\mathrm{E}_{A \sim \mathcal{U}_V}[|A_i|] = s \cdot \frac{|V_i|}{n}$.

Let $\beta_i = 3^{i-N-2}, \theta_i = (3\kappa)^{i-N-1}$. By Chernoff bound and our setting of s which satisfy that $s \geq \Omega(\frac{1}{(\delta\theta_i)^2 \beta_i})$, we have the following claim.

▶ **Claim 6.** *For any $i \leq N$, $\Pr_{A \sim \mathcal{U}_V}[|\frac{|A_i|}{s} - \frac{|V_i|}{n}| \leq \delta\theta_i] \geq 1 - \beta_i$.*

We assume for now that A is a fixed set with s vertices. We let $\sigma \sim \mathcal{D}$ denote that the edge ordering σ is sampled from $\mathcal{D}$. For any $v \in A$, let $Z_{v,i}$ be the indicator random variable of the event that the observed k-disc $\mathrm{disc}_k(v, \sigma)$ of v is isomorphic to Δ_i for $\sigma \sim \mathcal{D}$. Note that $\Pr_{\sigma \sim \mathcal{D}}[Z_{v,i} = 1] = \lambda(\Delta_i|\Delta_j)$ if $\mathrm{disc}_{k,G}(v) \cong \Delta_j$. Let $Y_i^{(\sigma)} := \frac{|\{v : v \in A, \mathrm{disc}_k(v,\sigma) \cong \Delta_i\}|}{s}$ denote the fraction of vertices in A with observed k-disc isomorphic to Δ_i. By definition, it holds that $Y_i^{(\sigma)} = \frac{1}{s} \sum_{\substack{v \in A_j \\ j \in \mathcal{G}(i) \cup \{i\}}} Z_{v,i}$, and furthermore, $\mathrm{E}_{\sigma \sim \mathcal{D}}[Y_i^{(\sigma)}] = \frac{1}{s} \sum_{j \in \mathcal{G}(i) \cup \{i\}} |A_j| \cdot \lambda(\Delta_i|\Delta_j)$.
Let $X_i^{(\sigma)} = (Y_i^{(\sigma)} - \sum_{j \in \mathcal{G}(i)} X_j^{(\sigma)} \cdot \lambda(\Delta_i|\Delta_j)) \cdot \lambda^{-1}(\Delta_i|\Delta_i)$.

We have the following claim.

▶ **Claim 7.** *For any $i \leq N$, it holds that $E_{\sigma \sim \mathcal{D}}[X_i^{(\sigma)}] = \frac{|A_i|}{s}$.*

Proof. We prove the claim by induction. For $i = 1$, it holds that $\mathrm{E}_{\sigma \sim \mathcal{D}}[X_1^{(\sigma)}] = \mathrm{E}_{\sigma \sim \mathcal{D}}[Y_1^{(\sigma)}] \cdot \lambda^{-1}(\Delta_1|\Delta_1) = \frac{|A_1|}{s} \cdot \lambda(\Delta_1|\Delta_1) \cdot \lambda^{-1}(\Delta_1|\Delta_1) = \frac{|A_1|}{s}$. Assuming that the claim holds for $i-1$, and we prove it holds for i as well. By definition, we have that

$$\begin{aligned}
&\mathrm{E}_{\sigma \sim \mathcal{D}}[X_i^{(\sigma)}] = \mathrm{E}_{\sigma \sim \mathcal{D}}[(Y_i^{(\sigma)} - \sum_{j \in \mathcal{G}(i)} X_j^{(\sigma)} \cdot \lambda(\Delta_i|\Delta_j)) \cdot \lambda^{-1}(\Delta_i|\Delta_i)] \\
= &\Big(\sum_{j \in \mathcal{G}(i) \cup \{i\}} \frac{|A_j|}{s} \cdot \lambda(\Delta_i|\Delta_j) - \sum_{j \in \mathcal{G}(i)} \mathrm{E}_{\sigma \sim \mathcal{D}}[X_j^{(\sigma)}] \cdot \lambda(\Delta_i|\Delta_j) \Big) \cdot \lambda^{-1}(\Delta_i|\Delta_i) \\
= &\Big(\sum_{j \in \mathcal{G}(i) \cup \{i\}} \frac{|A_j|}{s} \cdot \lambda(\Delta_i|\Delta_j) - \sum_{j \in \mathcal{G}(i)} \frac{|A_j|}{s} \cdot \lambda(\Delta_i|\Delta_j) \Big) \cdot \lambda^{-1}(\Delta_i|\Delta_i) = \frac{|A_i|}{s},
\end{aligned}$$

where the second to last equation follows from the induction. ◀

We can now bound the variance of $Y_i^{(\sigma)}$ as shown in the following claim.

▶ **Claim 8.** *For any $i \leq N$, it holds that $Var_{\sigma\sim\mathcal{D}}[Y_i^{(\sigma)}] \leq \frac{1}{s^2} \cdot d^{2k+1} \sum_{j\in\mathcal{G}(i)\cup\{i\}} |A_j| \cdot \lambda(\Delta_i|\Delta_j)$.*

Proof. Recall that $Y_i^{(\sigma)} = \frac{1}{s} \sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i}$. Note that for each $v \in A$, by the independence assumption on $\mathcal{D}$, the random variable $Z_{v,i}$ can only correlate with the corresponding variables for vertices that are within distance at most $2k$ from v. The number of such vertices is at most $1 + d + d^2 + \cdots + d^{2k} < d^{2k+1}$. Let $\mathrm{dt}(u, v)$ denote the distance between u, v in the graph G. Then we have that

$$\begin{aligned}
&\mathrm{E}_{\sigma\sim\mathcal{D}}[(\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i})^2] = \mathrm{E}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} \sum_{\substack{u\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i} \cdot Z_{u,i}] \\
= \;& \mathrm{E}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} (\sum_{\substack{u\in A_j \\ j\in\mathcal{G}(i)\cup\{i\} \\ \mathrm{dt}_G(u,v)\leq 2k}} Z_{v,i} \cdot Z_{u,i} + \sum_{\substack{u\in A_j \\ j\in\mathcal{G}(i)\cup\{i\} \\ \mathrm{dt}_G(u,v)>2k}} Z_{v,i} \cdot Z_{u,i})] \\
\leq \;& \mathrm{E}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} \sum_{\substack{u\in A_j \\ j\in\mathcal{G}(i)\cup\{i\} \\ \mathrm{dt}_G(u,v)\leq 2k}} Z_{v,i}] + \left(\sum_{j\in\mathcal{G}(i)\cup\{i\}} [|A_j|] \cdot \lambda(\Delta_i|\Delta_j)\right)^2 \\
\leq \;& d^{2k+1}\mathrm{E}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i}] + (\mathrm{E}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i}])^2 \\
= \;& d^{2k+1} \cdot \sum_{j\in\mathcal{G}(i)\cup\{i\}} |A_j| \cdot \lambda(\Delta_i|\Delta_j) + (\mathrm{E}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i}])^2,
\end{aligned}$$

where the first inequality follows from the fact that $Z_{u,i} \leq 1$, and that for any two vertices u, v with $\mathrm{dt}(u, v) > 2k$, $Z_{u,i}, Z_{v,i}$ are independent.

Then we have that

$$\begin{aligned}
\mathrm{Var}_{\sigma\sim\mathcal{D}}[Y_i^{(\sigma)}] &= \frac{1}{s^2} \cdot \mathrm{Var}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i}] \\
&= \frac{1}{s^2}\left(\mathrm{E}_{\sigma\sim\mathcal{D}}[(\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i})^2] - (\mathrm{E}_{\sigma\sim\mathcal{D}}[\sum_{\substack{v\in A_j \\ j\in\mathcal{G}(i)\cup\{i\}}} Z_{v,i}])^2\right) \\
&\leq \frac{1}{s^2} \cdot d^{2k+1} \sum_{j\in\mathcal{G}(i)\cup\{i\}} |A_j| \cdot \lambda(\Delta_i|\Delta_j).
\end{aligned}$$

◀

We next prove that each $X_i^{(\sigma)}$ is concentrated around its expectation with high probability.

▶ **Claim 9.** *For any $i \leq N$, it holds that $\Pr_{\sigma\sim\mathcal{D}}[|X_i^{(\sigma)} - E_{\sigma\sim\mathcal{D}}[X_i^{(\sigma)}]| \leq \theta_i\delta] \geq 1 - \beta_i$.*

Proof. We prove the claim by induction. For $i = 1$, it holds that

$$\begin{aligned}
&\Pr_{\sigma\sim\mathcal{D}}[|X_1^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[X_1^{(\sigma)}]| \leq \theta_1\delta] \leq \Pr_{\sigma\sim\mathcal{D}}[|Y_1^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[Y_1^{(\sigma)}]| \cdot \lambda^{-1}(\Delta_1|\Delta_1) \geq \delta\theta_1] \\
\leq \;& \frac{\mathrm{Var}_{\sigma\sim\mathcal{D}}[Y_1^{(\sigma)}]}{(\delta\theta_1)^2 \cdot \lambda^2(\Delta_1|\Delta_1)} \leq \frac{d^{2k+1}|A_1| \cdot \lambda(\Delta_1|\Delta_1)}{s^2 \cdot (\delta\theta_1)^2 \cdot \lambda^2(\Delta_1|\Delta_1)} \leq \frac{d^{2k+1}}{s(\delta\theta_1)^2 \cdot \lambda(\Delta_1|\Delta_1)} \leq \beta_1,
\end{aligned}$$

where the last inequality follows from our choice of β_1, θ_1 and s which satisfy that $s \geq \frac{d^{2k+1}}{(\delta\theta_1)^2\beta_1 \cdot \lambda(\Delta_1|\Delta_1)}$. Now let us consider arbitrary $i \geq 2$, assuming that the claim holds for any $j \leq i-1$. First, with probability (over the randomness that $\sigma \sim \mathcal{D}$) at least $1 - \sum_{j=1}^{i-1} \beta_j = 1 - \sum_{j=1}^{i-1} 3^{j-N-2} \geq 1 - \frac{\beta_i}{2}$, it holds that for all $j \leq i-1$, $|X_j^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[X_j^{(\sigma)}]| \leq \theta_j\delta$. This further implies that with probability at least $1 - \frac{\beta_i}{2}$,

$$\begin{aligned}
&|\sum_{j\in\mathcal{G}(i)} X_j^{(\sigma)} \cdot \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[(\sum_{j\in\mathcal{G}(i)} X_j^{(\sigma)} \cdot \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)})]| \\
\leq &\sum_{j\in\mathcal{G}(i)} |X_j^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[X_j^{(\sigma)}]| \cdot \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)} \\
\leq &\sum_{j\in\mathcal{G}(i)} \delta\theta_j \cdot \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)} \leq \kappa \cdot \sum_{j\in\mathcal{G}(i)} \delta\theta_j \leq \kappa \cdot \sum_{j=1}^{i-1} \delta(3\kappa)^{j-N} \leq \frac{\theta_i\delta}{2}.
\end{aligned}$$

Now note that

$$\begin{aligned}
&\Pr_{\sigma\sim\mathcal{U}}[|Y_i^{(\sigma)} - \mathrm{E}[Y_i^{(\sigma)}]| \cdot \lambda(\Delta_i|\Delta_i)^{-1} \geq \frac{\theta_i\delta}{2}] \leq \frac{4 \cdot \mathrm{Var}_{\sigma\sim\mathcal{D}}[Y_i^{(\sigma)}]}{(\delta\theta_i)^2 \cdot \lambda(\Delta_i|\Delta_i)^2} \\
\leq &\frac{4 \cdot d^{2k+1} \sum_{j\in\mathcal{G}(i)\cup\{i\}} |A_j| \cdot \lambda(\Delta_i|\Delta_j)}{s^2 \cdot (\delta\theta_i)^2 \cdot \lambda(\Delta_i|\Delta_i)^2} \leq \frac{4 \cdot d^{2k+1} \cdot \kappa}{s \cdot (\delta\theta_i)^2 \cdot \lambda(\Delta_i|\Delta_i)} \leq \frac{\beta_i}{2},
\end{aligned}$$

where the last inequality follows from our choice of β_i, θ_i and s which satisfy that $s \geq \frac{8\kappa \cdot d^{2k+1}}{(\delta\theta_i)^2\beta_i \cdot \lambda(\Delta_i|\Delta_i))}$.

Therefore, with probability (over $\sigma \sim \mathcal{D}$) at least $1 - \frac{\beta_i}{2} - \frac{\beta_i}{2} = 1 - \beta_i$, it holds that

$$\begin{aligned}
&|X_i^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[X_i^{(\sigma)}]| \\
= &\left|\frac{Y_i^{(\sigma)} - \sum_{j\in\mathcal{G}(i)} X_j^{(\sigma)} \cdot \lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)} - \mathrm{E}_{\sigma\sim\mathcal{D}}\left[\frac{Y_i^{(\sigma)} - \sum_{j\in\mathcal{G}(i)} X_j^{(\sigma)} \cdot \lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)}\right]\right| \\
= &\left|\frac{(Y_i^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[Y_i^{(\sigma)}])}{\lambda(\Delta_i|\Delta_i)} - \left(\sum_{j\in\mathcal{G}(i)} X_j^{(\sigma)} \cdot \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[(\sum_{j\in\mathcal{G}(i)} X_j^{(\sigma)} \cdot \frac{\lambda(\Delta_i|\Delta_j)}{\lambda(\Delta_i|\Delta_i)})]\right)\right| \\
\leq &\frac{\delta\theta_i}{2} + \frac{\delta\theta_i}{2} = \delta\theta_i.
\end{aligned}$$

◀

Now with probability (over both $A \sim \mathcal{U}_V$ and $\sigma \sim \mathcal{D}$) at least $1 - \beta_i - \beta_i$, it holds that

$$\begin{aligned}
&\left|X_i^{(\sigma)} - \frac{|V_i|}{n}\right| \leq \left|X_i^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[X_i^{(\sigma)}]\right| + \left|\mathrm{E}_{\sigma\sim\mathcal{D}}[X_i^{(\sigma)}] - \frac{|V_i|}{n}\right| \\
= &\left|X_i^{(\sigma)} - \mathrm{E}_{\sigma\sim\mathcal{D}}[X_i^{(\sigma)}]\right| + \left|\frac{|A_i|}{s} - \frac{|V_i|}{n}\right| \leq \delta\theta_i + \delta\theta_i = 2\delta\theta_i.
\end{aligned}$$

Finally, with probability at least $1 - 2\sum_{j=1}^{N} \beta_j = 1 - 2\sum_{j=1}^{N} 3^{j-N-2} \geq 1 - \frac{1}{3}$, it holds that for all $i \leq N$, $|X_i - \frac{|V_i|}{n}| \leq 2\theta_i\delta \leq \delta$. This completes the proof of the lemma. ◀

4 Constant-Query Property Testing

In this section, we show how to transform constant-query property testers in the adjacency list model to constant-space property testers in the random order stream model in a single pass and prove our main result Theorem 1. (Our transformation also works in the locally

random order model as defined in Definition 3, but for simplicity, we only state our result in the uniformly random order model.)

▶ **Definition 10.** Let $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ be a property of d-bounded graphs, where Π_n is a property of graphs with n vertices. We say that Π is testable with query complexity q, if for every ε, d and n, there exists an algorithm that performs $q = q(n, \varepsilon, d)$ queries to the adjacency list of the graph, and with probability at least $2/3$, accepts any n-vertex d-bounded graph G satisfying Π, and rejects any n-vertex d-bounded graph that is ε-far from satisfying Π. If $q = q(\varepsilon, d)$ is a function independent of n, then we call Π constant-query testable.

Similarly, we can define constant-space testable properties in graph streams.

▶ **Definition 11.** Let $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ be a property of d-bounded graphs, where Π_n is a property of graphs with n vertices. We say that Π is testable with space complexity q, if for every ε, d and n, there exists an algorithm that performs a single pass over an edge stream of an n-vertex d-bounded graph G, uses $q = q(n, \varepsilon, d)$ space, and with probability at least $2/3$, accepts G if it satisfies Π, and rejects G if it is ε-far from satisfying Π. If $q = q(\varepsilon, d)$ is a function independent of n, then we call Π constant-space testable.

The proof of Theorem 1 is based on the following known fact: every constant-query property tester can be simulated by some canonical tester which only samples a constant number of vertices, and explores the k-discs of these vertices, and then makes deterministic decisions based on the explored subgraph. This implies that it suffices to approximate the distribution of k-disc types of the input graph to test the corresponding property. Formally, we will use the following lemma relating the constant-time testable properties and their k-disc distributions. For any graph G, let $S_{G,k}$ denote the subgraph spanned by the union of k-discs rooted at k uniformly sampled vertices from G. The following lemma is implied by Lemma 3.2 in [7] (which was built on [14] and [13]). (The result in [7] is stated for d-bounded directed graphs, while it also holds in the undirected case.)

▶ **Lemma 12.** *Let $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ be any d-bounded graph property that is testable with $q = q(\varepsilon, d)$ query complexity in the adjacency list model. Then there exist integer n_0, $k = c \cdot q$ for some large universal constant c, and an infinite sequence of $\mathcal{F} = \{\mathcal{F}_n\}_{n\geq n_0}$ such that for any $n \geq n_0$, $\mathcal{F}_n$ is a set of digraphs, each being a union of k disjoint k-discs, and for any n-vertex graph G,*

- *if G satisfies Π_n, then with probability* at most $\frac{5}{12}$, *$S_{G,k}$ is isomorphic to one of the members in $\mathcal{F}_n$.*
- *if G is ε-far from satisfying Π_n, then with probability* at least $\frac{7}{12}$, *$S_{G,k}$ is isomorphic to one of the members in $\mathcal{F}_n$.*

Now we are ready to prove Theorem 1.

Proof Sketch of Theorem 1. The proof follows almost directly from the proof of Theorem 1.1 in [7]. We sketch the algorithm and its analysis, and refer to [7] for further details.

The algorithm is as follows. For any property $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ that is testable with query complexity $q = q(\varepsilon, d)$ in the adjacency list model, we set $k = c \cdot q$ as guaranteed in Lemma 12, and set $N = N(d, k)$, $\delta = \frac{1}{48(2kN)^k}$. Let G be any n-vertex graph with $n \geq n_1 := n_1(d, k)$ that is represented by a uniformly random order edge stream, where n_1 is some sufficiently large constant. (For graphs with $n < n_1$ vertices, one can trivally test Π_n with constant space.) Let Λ_k be the set of probabilities as guaranteed in Lemma 4. We first invoke the algorithm k-Disc_Distribution(Stream(G), $\Lambda_k, n, d, k, \delta$) to get estimators $X_1, \cdots, X_N$ for the

fraction $f_1, \cdots, f_N$ of vertices whose k-discs are isomorphic to $\Delta_1, \cdots, \Delta_N$, respectively. As guaranteed by Lemma 5, with probability at least $2/3$, it holds that for any $i \leq N$, $|X_i - f_i| \leq \delta$. Conditioned on this event, we approximate the frequency of each subgraph F in $\mathcal{F}_n$ as guaranteed in Lemma 12, where $F = (\Gamma_1, \cdots, \Gamma_k)$ is a multiset of k-discs. That is, for each $F = (\Gamma_1, \cdots, \Gamma_k)$, we calculate its empirical frequency as $\text{estim}(F) = \frac{\prod_{i=1}^{N} \binom{X_i \cdot n}{x_i}}{\binom{n}{k}}$, where x_i is the number of copies among $\Gamma_1, \cdots, \Gamma_k$ that are of the same type as Δ_i, for $1 \leq i \leq N$. Finally, we accept the graph if and only if $\sum_{F \in \mathcal{F}_n} \text{estim}(F) < \frac{1}{2}$.

By Lemma 5 and our setting of δ, the space used by the algorithm is $O(\frac{\kappa^{2N} \cdot d^{3k+2} \cdot 3^{3N+1}}{\delta^2 \lambda_{\min}}) = O(\frac{\kappa^{2N} \cdot d^{3k+2} \cdot 3^{3N+1} \cdot (2kN)^{2k}}{\lambda_{\min}})$.

For the correctness of the algorithm, note that if X_i's are good estimators for f_i's, then the empirical probability $\text{estim}(F)$ is close to the probability that $S_{G,k}$, the subgraph spanned by the union of k-discs rooted at k uniformly sampled vertices from G, spans a subgraph that is isomorphic to F. This implies that the quantity $\sum_{F \in \mathcal{F}_n} \text{estim}(F)$ is a good estimator for the probability that $S_{G,k}$ is isomorphic to one of the members in $\mathcal{F}$. Combining this with Lemma 12, we can show that if G satisfies Π_n, then $\sum_{F \in \mathcal{F}_n} \text{estim}(F) < \frac{1}{2}$ and if G is ε-far from satisfying the property Π_n, then $\sum_{F \in \mathcal{F}_n} \text{estim}(F) \geq \frac{1}{2}$. We omit details here. ◀

5 Constant-Time Approximation Algorithms

As we mentioned in the introduction, to simulate any constant-time algorithm that is independent of the labeling of the vertices, and accesses the graph by sampling random vertices and exploring neighborhoods (or k-discs for some k) of these vertices, it suffices to have the distribution of k-disc types. Now we explain slightly more about this simulation and sketch the proof of Theorem 2. In order to approximate the size of the solution of an optimization problem (e.g., maximum matching, minimum vertex cover), it has been observed by Parnas and Ron [28] that it suffices to have efficient oracle $\mathcal{O}_S$ access to a solution S. This is true since one can attain a good estimator for the size of S by sampling a constant number of vertices, performing corresponding queries to the oracle $\mathcal{O}_S$ and then returning the fraction of vertices that belong to S based on the returned answers from $\mathcal{O}_S$. Nguyen and Onak [26] implemented such an oracle via an elegant approach of locally simulating the classical greedy algorithm. In particular, they showed the following result.

▶ **Lemma 13** ([26]). *There exist $q = q(\varepsilon, d)$, an oracle $\mathcal{O}_M$ to a maximal matching M, and an algorithm that queries $\mathcal{O}_M$ about all the edges incident to a set of $s = O(1/\varepsilon^2)$ randomly sampled vertices and with probability at least $2/3$, returns an estimator that is $(1, \varepsilon n)$-approximation of the size of M, and each query to $\mathcal{O}_M$ performs at most q queries to the adjacency list of the graph.*

A key observation is that the algorithm in Lemma 13 can be viewed as first sampling s q-discs from the graph and then perform $\mathcal{O}_M$ queries on each of these q-discs. It is easy to see that with high probability 0.99, all these q-discs are disjoint. Furthermore, the answer of the above oracle only depends on the structure of the corresponding neighborhood of the starting vertex v and the random ordering of the edges belonging to this neighborhood. Now we can approximate the size of a maximal matching in the random order streaming model as follows: we first invoke Algorithm 2 to get an estimator for the distribution of q-discs. Then we can simulate the oracle on this distribution.

Acknowledgments. We would like to thank G. Cormode, H. Jowhari for helpful discussions.

References

1. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.
2. Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1723–1742. SIAM, 2017.
3. Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. *Advances in mathematics*, 223(6):2200–2218, 2010.
4. Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms-ESA 2015*, pages 263–274. Springer, 2015.
5. Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 641–650. ACM, 2008.
6. Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005.
7. Artur Czumaj, Pan Peng, and Christian Sohler. Relating two property testing models for bounded degree directed graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2016, pages 1033–1045. ACM, 2016.
8. Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.
9. Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM Journal on Computing*, 38(5):1709–1727, 2008.
10. Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. Testing and spot-checking of data streams. *Algorithmica*, 34(1):67–80, 2002.
11. Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.
12. Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32:302–343, 2002.
13. Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40(2):534–566, 2011.
14. Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Structures & Algorithms*, 23(1):23–57, 2003.
15. Avinatan Hassidim, Jonathan A Kelner, Huy N Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 22–31. IEEE, 2009.
16. Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In *External Memory Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, May 20-22, 1998*, pages 107–118, 1998.
17. Zengfeng Huang and Pan Peng. Dynamic graph stream algorithms in o(n) space. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 18:1–18:16, 2016.
18. Hiro Ito, Shin-Ichi Tanigawa, and Yuichi Yoshida. Constant-time algorithms for sparsity matroids. In *International Colloquium on Automata, Languages, and Programming*, pages 498–509. Springer, 2012.
19. Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751. Society for Industrial and Applied Mathematics, 2014.

20 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating max-cut. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1263–1282. Society for Industrial and Applied Mathematics, 2015.

21 Ken-ichi Kawarabayashi and Yuichi Yoshida. Testing subdivision-freeness: property testing meets structural graph theory. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 437–446. ACM, 2013.

22 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 231–242. Springer, 2012.

23 Sharon Marko and Dana Ron. Approximating the distance to properties in bounded-degree and general sparse graphs. *ACM Transactions on Algorithms (TALG)*, 5(2):22, 2009.

24 Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.

25 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013.

26 Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 327–336. IEEE, 2008.

27 Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1123–1131. SIAM, 2012.

28 Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1):183–196, 2007.

29 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

30 Xiaoming Sun and David P Woodruff. Tight bounds for graph problems in insertion streams. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 40. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

31 Shin-Ichi Tanigawa and Yuichi Yoshida. Testing the supermodular-cut condition. *Algorithmica*, 71(4):1065–1075, 2015.

32 Elad Verbin and Wei Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 11–25. SIAM, 2011.

33 Yuichi Yoshida and Hiro Ito. Property testing on k-vertex-connectivity of graphs. In *Automata, Languages and Programming*, pages 539–550. Springer, 2008.

34 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM J. Comput.*, 41(4):1074–1093, 2012.

Deterministic Graph Exploration with Advice

Barun Gorain[1] **and Andrzej Pelc**[*2]

1 **Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec, Canada**
baruniitg123@gmail.com

2 **Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec, Canada**
pelc@uqo.ca

Abstract

We consider the task of graph exploration. An n-node graph has unlabeled nodes, and all ports at any node of degree d are arbitrarily numbered $0, \ldots, d-1$. A mobile agent has to visit all nodes and stop. The exploration time is the number of edge traversals. We consider the problem of how much knowledge the agent has to have a priori, in order to explore the graph in a given time, using a deterministic algorithm. This a priori information (advice) is provided to the agent by an oracle, in the form of a binary string, whose length is called the size of advice. We consider two types of oracles. The instance oracle knows the entire instance of the exploration problem, i.e., the port-numbered map of the graph and the starting node of the agent in this map. The map oracle knows the port-numbered map of the graph but does not know the starting node of the agent. What is the minimum size of advice that must be given to the agent by each of these oracles, so that the agent explores the graph in a given time?

We first consider exploration in polynomial time, and determine the exact minimum size of advice to achieve it. This size is $\log\log\log n - \Theta(1)$, for both types of oracles.

When advice is large, there are two natural time thresholds: $\Theta(n^2)$ for a map oracle, and $\Theta(n)$ for an instance oracle, that can be achieved with sufficiently large advice. We show that, with a map oracle, time $\Theta(n^2)$ cannot be improved in general, regardless of the size of advice. We also show that the smallest size of advice to achieve this time is larger than n^δ, for any $\delta < 1/3$.

For an instance oracle, advice of size $O(n \log n)$ is enough to achieve time $O(n)$. We show that, with any advice of size $o(n \log n)$, the time of exploration must be at least n^ϵ, for any $\epsilon < 2$, and with any advice of size $O(n)$, the time must be $\Omega(n^2)$.

We also investigate minimum advice sufficient for fast exploration of Hamiltonian graphs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases algorithm, graph, exploration, mobile agent, advice

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.132

1 Introduction

Exploration of networks by visiting all of their nodes is one of the basic tasks performed by a mobile agent in networks. In applications, a software agent may need to collect data placed at nodes of a network, or a mobile robot may need to collect samples of air or ground in a contaminated mine whose corridors form links of a network, with corridor crossings represented by nodes.

* Research supported in part by NSERC Discovery Grant 8136 – 2013 and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 132; pp. 132:1–132:14

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

The network is modeled as a simple connected undirected graph $G = (V, E)$ with n nodes, called *graph* in the sequel. Nodes are unlabeled, and all ports at any node of degree d are arbitrarily numbered $0, \ldots, d-1$. The agent is initially situated at a starting node v of the graph. When the agent located at a current node u gets to a neighbor w of u by taking port i, it learns the port j by which it enters node w and it learns the degree of w. The agent has to visit all nodes of the graph and stop. The agent is computationally unbounded and cannot mark the visited nodes.

The *time* of the exploration is the number of edge traversals. We consider the problem of how much knowledge the agent has to have a priori, in order to explore the graph in a given time, using a deterministic algorithm. It is well-known that some information is necessary, as witnessed even by the class of rings in which ports at all nodes are numbered 0,1 in clockwise order. Navigating in such a ring, the agent cannot learn its size. If there existed an exploration algorithm not using any a priori knowledge, then it would have to stop after some t steps in every ring, and hence would fail to explore a $(t+2)$-node ring.

Following the paradigm of *algorithms with advice* [1, 13, 17, 18, 21, 20, 22, 23, 24, 25, 26, 27, 29], this a priori information (advice), needed for exploration, is provided to the agent by an *oracle*, in the form of a binary string, whose length is called the *size of advice*. We consider two types of oracles. An *instance oracle* knows the entire instance of the exploration problem, i.e., the port-numbered map of the graph and the starting node of the agent in this map. A *map oracle* knows the port-numbered map of the graph but does not know the starting node of the agent. Formally, a map oracle is a function $f : \mathcal{G} \longrightarrow S$, where $\mathcal{G}$ is the set of graphs and S is the set of finite binary strings. An instance oracle is a function $f : \mathcal{I} \longrightarrow S$, where $\mathcal{I}$ is the set of pairs (G, v), with $G \in \mathcal{G}$ and v being the starting node of the agent in graph G. The advice s is an input to an exploration algorithm. We say that exploration in time t with advice of size x given by an instance oracle is possible, if there exists advice of length x depending on the instance (G, v), and an exploration algorithm using this advice, which explores every graph in time t, starting from node v. Likewise, we say that exploration in time t with advice of size x given by a map oracle is possible, if there exists advice of length x depending on the graph G, and an exploration algorithm using this advice, which explores every graph in time t, starting from any node. (Integers x and t depend on the size of the graph.) Proving that such an exploration is possible consists in showing an oracle of the appropriate type giving advice of size x, and an exploration algorithm using this advice and working in time t, for any graph and any starting node. Proving that such an exploration is impossible consists in showing that, for any oracle of the appropriate type giving advice of size x, and for any exploration algorithm using it, there exists a graph and a starting node for which this algorithm will exceed time t.

The central question studied in this paper is:

What is the minimum size of advice that has to be given to the agent by an instance oracle (resp. by a map oracle) to permit the agent to explore any graph in a given time?

Our main contributions are negative results of two types:

- impossibility results showing that the less powerful map oracle cannot help to achieve the same exploration time as the more powerful instance oracle, regardless of the size of advice;
- lower bounds showing that the size of some natural advice leading to a simple exploration in a given time cannot be improved significantly.

While in most cases our bounds on the size of advice are asymptotically tight, in one case the remaining gap is cubic.

1.1 Our results

We first consider exploration in polynomial time, and determine the exact minimum size of advice to achieve it. Indeed, we prove that some advice of size $\log\log\log n - c$, for any constant c, is sufficient to permit polynomial exploration of all n-node graphs, and that no advice of size $\log\log\log n - \phi(n)$, where ϕ is any function diverging to infinity, can help to do this. Both these results hold both for the instance and for the map oracles.

On the other side of the spectrum, when advice is large, there are two natural time thresholds: $\Theta(n^2)$ for a map oracle, and $\Theta(n)$ for an instance oracle. This is because, in both cases, these time benchmarks can be achieved with sufficiently large advice (advice of size $O(n\log n)$ suffices). We show that, with a map oracle, time $\Theta(n^2)$ cannot be improved in general, regardless of the size of advice. What is then the smallest advice to achieve time $\Theta(n^2)$ with a map oracle? We show that this smallest size of advice is larger than n^δ, for any $\delta < 1/3$.

For large advice, the situation changes significantly when we allow an instance oracle instead of a map oracle. In this case, advice of size $O(n\log n)$ is enough to achieve time $O(n)$. Is such a large advice needed to achieve linear time? We answer this question affirmatively. Indeed, we show more: with any advice of size $o(n\log n)$, the time of exploration must be at least n^ϵ, for any $\epsilon < 2$, and with any advice of size $O(n)$, the time must be $\Omega(n^2)$.

We finally look at Hamiltonian graphs, as for them it is possible to achieve the absolutely optimal exploration time $n-1$, when sufficiently large advice (of size $O(n\log n)$) is given by an instance oracle. We show that a map oracle cannot achieve this: regardless of the size of advice, the time of exploration must be $\Omega(n^2)$, for some Hamiltonian graphs. On the other hand, even for an instance oracle, with advice of size $o(n\log n)$, optimal time $n-1$ cannot be achieved: indeed, we show that the time of exploration with such advice must sometimes exceed the optimal time $n-1$ by a summand n^ϵ, for any $\epsilon < 1$.

Our results permit us to compare advice of different size and of different quality. The size is defined formally, and for quality we may say that advice given by an instance oracle is superior to advice given by a map oracle, because an instance oracle, seeing not only the graph but also the starting node of the agent, can use the allowed bits of advice in a better way. Looking from this perspective it turns out that both size and quality of advice provably matter. The fact that quality of advice matters is proved by the following pair of results: for a map oracle, time $\Theta(n^2)$ cannot be beaten, regardless of the size of advice, while for an instance oracle time $O(n)$ can be achieved with $O(n\log n)$ bits of advice. The fact that the size of advice matters (with the same quality) is proved by the following pair of results: for an instance oracle, time $O(n)$ can be achieved with $O(n\log n)$ bits of advice, but with $o(n\log n)$ bits of advice time must be at least n^ϵ, for any $\epsilon < 2$.

1.2 Related work

The problem of exploration and navigation of mobile agents in an unknown environment has been extensively studied in the literature for many decades (cf. the survey [33]). The explored environment has been modeled in two distinct ways: either as a geometric terrain in the plane, e.g., an unknown terrain with convex obstacles [9], or a room with polygonal [12] or rectangular [5] obstacles, or as we do in this paper, i.e., as a graph, assuming that the agent may only move along its edges. The graph model can be further specified in two different ways: either the graph is directed, in which case the agent can move only from tail to head of a directed edge [2, 6, 7, 13], or the graph is undirected (as we assume) and the agent can traverse edges in both directions [4, 8, 16, 30, 31]. The efficiency measure

adopted in most papers dealing with exploration of graphs is the time (or cost) of completing this task, measured by the number of edge traversals by the agent. Some authors [4, 8, 16] impose further restrictions on the moves of the agent.

Another direction of research concerns exploration of anonymous graphs. In this case it is impossible to explore arbitrary graphs and stop after exploration, if no marking of nodes is allowed, and if nothing is known about the graph. Hence some authors [6, 7] allow *pebbles* which the agent can drop on nodes to recognize already visited ones, and then remove them and drop them in other places. A more restrictive scenario assumes a stationary token that is fixed at the starting node of the agent [11, 32]. Exploring anonymous graphs without the possibility of marking nodes (and thus possibly without stopping) is investigated, e.g., in [14, 19]. The authors concentrate attention not on the cost of exploration but on the minimum amount of memory sufficient to carry out this task. In the absence of marking nodes, in order to guarantee stopping after exploration, some knowledge about the graph is required, e.g., an upper bound on its size [11, 34].

Providing nodes or agents with arbitrary kinds of information that can be used to perform network tasks more efficiently has been previously proposed in [1, 13, 17, 18, 21, 20, 22, 23, 24, 25, 26, 27, 29] in contexts ranging from graph coloring to broadcasting and leader election. This approach was referred to as *algorithms with advice*. The advice is given either to nodes of the network or to mobile agents performing some network task. In the first case, instead of advice, the term *informative labeling schemes* is sometimes used, if different nodes can get different information.

Several authors studied the minimum size of advice required to solve network problems in an efficient way. In [27], given a distributed representation of a solution for a problem, the authors investigated the number of bits of communication needed to verify the legality of the represented solution. In [21], the authors compared the minimum size of advice required to solve two information dissemination problems using a linear number of messages. In [22], it was shown that advice of constant size given to the nodes enables the distributed construction of a minimum spanning tree in logarithmic time. In [15, 17], the advice paradigm was used for online problems. In particular, in [15] the authors studied online graph exploration with advice in labeled weighted graphs. In the case of [29], the issue was not efficiency but feasibility: it was shown that $\Theta(n \log n)$ is the minimum size of advice required to perform monotone connected graph clearing. In [26], the authors studied radio networks for which it is possible to perform centralized broadcasting in constant time. In [24], the authors studied the problem of topology recognition with advice given to nodes. The topic of [28] and [35] was the size of advice needed for fast leader election, resp. in anonymous trees and in arbitrary anonymous graphs. Exploration with advice was previously studied only for trees [20], and algorithm performance was measured using the competitive approach. In the present paper, the performance measure of an algorithm is the order of magnitude of exploration time, and hence the case of trees is trivial, as they can be explored in linear time without any advice.

2 Exploration in polynomial time

As a warm-up, we first consider the following question: What is the minimum size of advice permitting the agent to explore any graph in time polynomial in the size of the graph? In this section we give the exact answer to this question, both for the instance oracle and for the map oracle.

It is well-known that, if the agent knows an upper bound n' on the number n of nodes of the graph, then exploration in time polynomial in n' is possible, starting from any node of the

graph. The first result implying this fact was proved in [3]. The exploration proposed there works in time $O(n'^5 \log n')$, and is based on Universal Traversal Sequences (UTS). Later on, an exploration algorithm working in time polynomial in n' based on Universal Exploration Sequences (UXS) was established in [34]. While the polynomial in the latter paper has much higher degree, the solution from [34] can be carried out in logarithmic memory. Both UTS and UXS permit to find a sequence of port numbers to be followed by the agent, regardless of the topology of the graph and of its starting node. In the case of UTS, the sequence of port numbers to be followed is the UTS itself, and in the case of UXS it is constructed term by term, on the basis of the UXS and of the port number by which the agent entered the current node. Regardless of which solution is used, we have the following proposition:

▶ **Proposition 1** ([3, 34]). *If the agent knows an upper bound n' on the number n of nodes of the graph, there exists an algorithm with input n' that permits the agent starting at any node of the graph to explore the graph and stop after $P(n')$ steps, where P is some polynomial.*

The positive part of our result on minimum advice is formulated in the following lemma. Its proof is based on Proposition 1. The advice given to the agent is some prefix of the binary representation of the number $\lfloor \log \log n \rfloor$, on the basis of which the agent computes a rough but sufficiently precise upper bound on the size of the graph which permits it to explore the graph, in time polynomial in its size.

▶ **Lemma 2.** *For any positive constant c, there exists an exploration algorithm using advice of size $\lfloor \log \log \log n - c \rfloor$, that works in time polynomial in n, for any n-node graph.*

The next result shows that the upper bound from the previous lemma is tight. Indeed, the following lower bound holds even for oriented rings, i.e., rings in which ports 0 and 1 are in clockwise order at every node.

▶ **Lemma 3.** *For any function $\phi : \mathbb{N} \to \mathbb{N}$ such that $\phi(n) \to \infty$ as $n \to \infty$, it is not possible to explore an n-node oriented ring in polynomial time, using advice of size at most $\log \log \log n - \phi(n)$.*

Notice that Lemmas 2 and 3 hold both for the instance oracle and for the map oracle. The positive result from Lemma 2 holds even for the map oracle, as the advice concerns the size of the graph and does not require knowing the starting node of the agent. The negative result from Lemma 3 holds even for the instance oracle, as it is true even in oriented rings, where knowledge of the starting node does not provide any insight, since all nodes look the same. Hence Lemmas 2 and 3 imply the following theorem that gives a precise answer to the question stated at the beginning of this section.

▶ **Theorem 4.** *The minimum size of advice permitting the agent to explore any graph in time polynomial in the size n of the graph is $\log \log \log n - \Theta(1)$, both for the instance oracle and for the map oracle.*

3 Fast exploration

When advice given to the agent can be large, there are two natural time thresholds: $\Theta(n^2)$ for a map oracle, and $\Theta(n)$ for an instance oracle. This is because, in both cases, these time benchmarks can be achieved with sufficiently large advice. Indeed, we have the following proposition.

▶ **Proposition 5.**

1. *There exists an exploration algorithm, working in time $O(n^2)$ and using advice of size $O(n \log n)$, provided by a map oracle, for n-node graphs.*
2. *There exists an exploration algorithm, working in time $O(n)$ and using advice of size $O(n \log n)$, provided by an instance oracle, for n-node graphs.*

In the rest of this section we prove negative results indicating the quality of the natural solution given in Proposition 5. For the map oracle, we show that quadratic exploration time cannot be beaten, and we give a lower bound on the size of advice sufficient to guarantee this time. For the instance oracle, we show that Proposition 5 gives optimal advice for linear exploration time.

3.1 Map oracle

Our first result for the map oracle shows that, regardless of the size of advice, exploration time $\Theta(n^2)$ cannot be beaten, for some n-node graphs.

We will use the following construction from [10] of a family $\mathcal{H}_X$ of graphs.

Let H be an $\frac{m}{2}$-regular graph with m nodes, where m is even, e.g., the complete bipartite graph. Let T be the set of edges of any spanning tree of H. Let S be the set of edges of H outside T. Let $s = |S| = \frac{m^2}{4} - m + 1$ and $S = \{e_1, e_2, \cdots, e_s\}$.

For $x \in \{0,1\}^s \setminus \{0^s\}$, the $(2m)$-node graph H_x is constructed from H by taking two disjoint copies H' and H'' of H, and crossing some pairs of edges from one copy to the other. For $i = 1, \cdots, s$, if the $i-$th bit of x is 1, then the edge $e_i = (u_i, v_i)$ is deleted from both copies of H and two copies of e_i are crossed between the two copies of H. More precisely, let $\{v_1, \cdots, v_m\}$ be the set of nodes of H and let v'_i and v''_i be the nodes corresponding to v_i, in H' and H'', respectively. Let V' and V'' be the sets of nodes of H' and H'', respectively. Define $H_x = (V' \cup V'', E_x)$, where $E_x = \{(v'_i, v'_j), (v''_i, v''_j) : (v_i, v_j) \in T\} \cup \{(v'_i, v''_j), (v''_i, v'_j) : e_k = (v_i, v_j) \in S \text{ and } x_k = 1\}$. Let $\mathcal{H}_X = \{H_x : x \in \{0,1\}^s \setminus \{0^s\}\}$.

According to the result from [10], for every node $v \in H$, there exists some sequence $x(v) \in \{0,1\}^s \setminus \{0^s\}$ such that if an exploration of H performed according to some sequence W of port numbers, starting from node v_1, visits node v at most s times, then in one of the copies H' or H'' in $H_{x(v)}$ the node v' or v'' is not visited at all, if the same sequence W is used to explore the graph $H_{x(v)}$ starting from v'_1. Intuitively, the result from [10] shows a class of graphs with the property that if some node in one of these graphs is not visited many times, then the exploration algorithm fails *in some other graph* of this class. There is no control in which graph of the class this will happen. We use the graphs from [10] as building blocks to prove a different kind of lower bound. Indeed, we construct a *single graph* having the property that if some of its nodes are not visited many times, then exploration must fail in this graph. This will prove a lower bound on exploration time for some graph, even if the agent knows the entire graph.

Using the graphs $H_x \in \mathcal{H}_X$ from [10] we construct the graph $\widehat{G}$ as follows. For any $1 \leq i \leq m$, let $v'_1(i)$ be the node corresponding to node v'_1 from H' in the graph $H_{x(v_i)}$. Connect the graphs $H_{x(v_i)}$, for $1 \leq i \leq m$, and an oriented cycle C with nodes $\{y_1, \cdots, y_m\}$ (port numbers 0,1 are in clockwise order at each node of the cycle), by adding edges $(y_i, v'_1(i))$, for $1 \leq i \leq m$. The port numbers corresponding to these edges are: 2 at y_i and $\frac{m}{2}$ at $v'_1(i)$. The cycle C is called the *main cycle* of $\widehat{G}$. See Fig. 1.

Let $n = 2m^2 + m$ be the number of nodes in $\widehat{G}$.

By the construction of $\widehat{G}$, any exploration algorithm with the agent starting from any node of the main cycle, has the following *obliviousness property*. For any step i of the

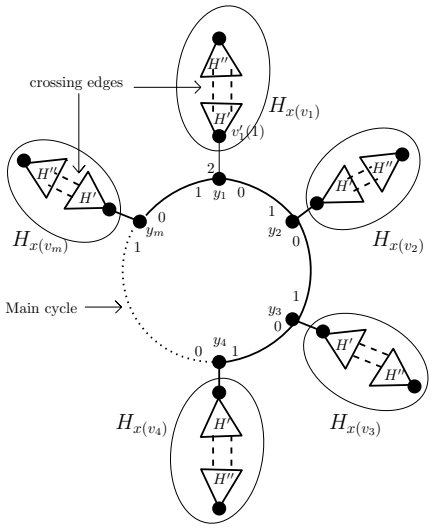


Figure 1 Construction of $\widehat{G}$.

algorithm, if the agent is at some node v in this step, and the algorithm prescribes taking some port p at this node, then the port q through which the agent enters the adjacent node w in the $(i+1)$-th step, and the degree of the node w are predetermined (i.e., they are independent of the starting node in the main cycle). Intuitively, the agent does not learn anything during the algorithm execution. Therefore, every exploration algorithm with the agent starting from any node of the main cycle can be uniquely coded by a sequence of port numbers which the agent takes in consecutive steps of its exploration.

Let $\mathcal{A}$ be any exploration algorithm for $\widehat{G}$, and suppose that the agent starts from some node of the main cycle. We use $\cdot$ for concatenation of sequences.

▶ **Lemma 6.** *Let U be the sequence of port numbers corresponding to the movement of the agent according to algorithm $\mathcal{A}$, starting at some node of the main cycle C of $\widehat{G}$. Then $U = B'_1 \cdot (2) \cdot B_1 \cdot (\frac{m}{2}) \cdot B'_2 \cdot (2) \cdot B_2 \cdot (\frac{m}{2}) \cdots B'_p \cdot (2) \cdot B_p \cdot (\frac{m}{2}) \cdot B'_{p+1}$, where each B'_j is a sequence of port numbers corresponding to the movement of the agent along C and each B_j is a sequence of port numbers corresponding to the movement of the agent inside some $H_{x(v_i)}$.*

Call an exploration algorithm of $\widehat{G}$ *non-repetitive*, if the agent, starting from the main cycle, enters each $H_{x(v_i)}$, for $1 \leq i \leq m$, exactly once. By definition, the sequence of port numbers corresponding to a non-repetitive algorithm can be written as $D'_1 \cdot (2) \cdot D_1 \cdot (\frac{m}{2}) \cdot D'_2 \cdot (2) \cdot D_2 \cdot (\frac{m}{2}) \cdots D'_m \cdot (2) \cdot D_m \cdot (\frac{m}{2}) \cdot D'_{m+1}$, where each D'_j is a sequence of port numbers corresponding to the movement of the agent along C and each D_j is a sequence of port numbers corresponding to the movement of the agent inside some $H_{x(v_i)}$. Notice that since the algorithm is non-repetitive, the number of blocks D_j is exactly m.

The following lemma proves that in order to show a lower bound on the exploration time in $\widehat{G}$, it is enough to consider only the class of non-repetitive algorithms.

▶ **Lemma 7.** *If the agent starts from some node of the main cycle of $\widehat{G}$ and executes any exploration algorithm $\mathcal{A}$ of $\widehat{G}$, then there exists a non-repetitive algorithm $\mathcal{A}'$ for this agent, such that the exploration time of $\mathcal{A}'$ is at most the exploration time of $\mathcal{A}$.*

The next lemma implies that the sequence U corresponding to a correct non-repetitive exploration algorithm must be long.

▶ **Lemma 8.** *Let $U = D'_1 \cdot (2) \cdot D_1 \cdot (\frac{m}{2}) \cdot D'_2 \cdot (2) \cdot D_2 \cdot (\frac{m}{2}) \cdots D'_m \cdot (2) \cdot D_m \cdot (\frac{m}{2}) \cdot D'_{m+1}$ be the sequence of port numbers corresponding to a non-repetitive algorithm. If there exists some D_i such that the agent following D_i in H starting from node v_1 visits some node v_j of H at most s times, then there exists a starting node in the main cycle of $\widehat{G}$, such that the agent starting at this node and following U does not visit all the nodes of $\widehat{G}$.*

▶ **Theorem 9.** *Any exploration algorithm using any advice given by a map oracle must take time $\Omega(n^2)$ on graph $\widehat{G}$, for some starting node in the main cycle, for arbitrarily large n.*

Proof. By Lemma 7, it is enough to consider only non-repetitive algorithms. Let $U = D'_1 \cdot (2) \cdot D_1 \cdot (\frac{m}{2}) \cdot D'_2 \cdot (2) \cdot D_2 \cdot (\frac{m}{2}) \cdots D'_m \cdot (2) \cdot D_m \cdot (\frac{m}{2}) \cdot D'_{m+1}$ be the sequence of port numbers corresponding to such an algorithm. Then by Lemma 8, for each i, $1 \leq i \leq m$, the agent following D_i in H starting from node v_1 visits each node v_j of H, for $1 \leq j \leq m$, at least $s+1$ times. Therefore the length of D_i is at least $(s+1)m$. Hence, the length of U is at least $\sum_{i=1}^{m}(s+1)m = (s+1)m^2 = m^2(\frac{m^2}{4} - m + 1)$. Since $n = 2m^2 + m$, the length of U is in $\Omega(n^2)$. ◀

Theorem 9 shows that, for some n-node graph, no advice given by a map oracle can help to explore this graph in time better than $\Theta(n^2)$. It is then natural to ask what is the minimum size of advice to achieve time $\Theta(n^2)$ with a map oracle, for every n-node graph. Our next result shows that any exploration algorithm using advice of size n^δ for $\delta < \frac{1}{3}$, must take time $\omega(n^2)$, on some n-node graph.

Fix a constant $\epsilon < \frac{1}{2}$. Let H be an $\frac{m}{2}$-regular graph with m nodes, where m is even. Let $\{v_1, \cdots, v_m\}$ be the set of nodes of H. Consider a subset $Z \subset \{1, 2, \cdots, m\}$ of size m^ϵ. Let $p = m^\epsilon$ and $n = 2mp + p$. We construct an n-node graph $\widehat{G_Z}$ from H. The construction of $\widehat{G_Z}$ is similar to the construction of $\widehat{G}$ at the beginning of this section. Let $Z = \{z_1, z_2, \cdots, z_p\}$. To construct $\widehat{G_Z}$, connect the (previously described) graphs $H_{x(v_{z_i})}$, for $1 \leq i \leq p$, and an oriented cycle C' (called the main cycle) with nodes $\{y_1, \cdots, y_p\}$, by adding edges $(y_i, v'_1(z_i))$, for $1 \leq i \leq p$. The port numbers corresponding to these edges are: 2 at y_i and $\frac{m}{2}$ at $v'_1(z_i)$. Note that the same obliviousness property applies to exploration algorithms in graphs $\widehat{G_Z}$, when the agent starts from a node of the main cycle.

Let $\widehat{\mathcal{G}_Z}$ be the set of all possible graphs $\widehat{G_Z}$ constructed from H. We have $|\widehat{\mathcal{G}_Z}| = \binom{m}{p}$.

▶ **Theorem 10.** *For any $\epsilon < \frac{1}{2}$, any exploration algorithm using advice of size $o(n^{\frac{\epsilon}{1+\epsilon}} \log n)$ must take time $\omega(n^2)$ on some graph of the class $\widehat{\mathcal{G}_Z}$ and for some starting node in the main cycle of this graph, for arbitrarily large n.*

Proof. Since $\epsilon < \frac{1}{2}$, there exists an integer c such that $\epsilon < \frac{c-1}{2c-1}$. We show that if the size of the advice is at most $\frac{1}{2c} m^\epsilon \log(m^{1-\epsilon}) \leq \frac{1-\epsilon}{2c(1+\epsilon)} (\frac{n}{2})^{\frac{\epsilon}{1+\epsilon}} \log \frac{n}{2}$, then there exists a graph in $\widehat{\mathcal{G}_Z}$ for which the time required for exploration is $\omega(n^2)$. We have $|\widehat{\mathcal{G}_Z}| = \binom{m}{m^\epsilon} \geq (m^{1-\epsilon})^{m^\epsilon}$. There are fewer than $(m^{1-\epsilon})^{\frac{m^\epsilon}{c}}$ different binary strings of length at most $\frac{1}{2c} m^\epsilon \log(m^{1-\epsilon})$. By the pigeonhole principle, there exists a family of graphs $\widehat{\mathcal{G}} \subset \widehat{\mathcal{G}_Z}$ of size at least $(m^{1-\epsilon})^{(c-1)\frac{m^\epsilon}{c}}$ such that all the graphs in $\widehat{\mathcal{G}}$ get the same advice.

Define $F(\widehat{\mathcal{G}}) = \bigcup \Big\{ \{v_{z_1}, v_{z_2}, \cdots, v_{z_p}\} : Z = \{z_1, z_2, \cdots, z_p\} \text{ and } \widehat{G_Z} \in \widehat{\mathcal{G}} \Big\}$. Intuitively, $F(\widehat{\mathcal{G}})$ is the subset of nodes of H, such that for each $v \in F(\widehat{\mathcal{G}})$, there exists some graph in $\widehat{\mathcal{G}}$ that contains $H_{x(v)}$ as a subgraph.

Claim: $|F(\widehat{\mathcal{G}})| \geq |\widehat{\mathcal{G}}|^{\frac{1}{p}}$.

Proof. We prove the claim by contradiction. Suppose that $|F(\widehat{\mathcal{G}})| < |\widehat{\mathcal{G}}|^{\frac{1}{p}}$. Each graph in $\widehat{\mathcal{G}}$ has p different subgraphs $H_{x(v)}$, where $v \in |F(\widehat{\mathcal{G}})|$. There are $\binom{|F(\widehat{\mathcal{G}})|}{p}$ different graphs in $\widehat{\mathcal{G}}$ which is at most $|F(\widehat{\mathcal{G}})|^p < |\widehat{\mathcal{G}}|$. This contradiction proves the claim. ◀

Consider the exploration of some graph $\widehat{G_Z} \in \widehat{\mathcal{G}}$ starting from the main cycle. Let $U = D'_1 \cdot (2) \cdot D_1 \cdot (\frac{m}{2}) \cdot D'_2 \cdot (2) \cdot D_2 \cdot (\frac{m}{2}) \cdots D'_p \cdot (2) \cdot D_m \cdot (\frac{m}{2}) \cdot D'_{p+1}$ be the sequence of port numbers corresponding to a non-repetitive algorithm exploring $\widehat{G_Z}$. Then for each i, $1 \leq i \leq p$, the agent following D_i in H starting from node v_1 must visit each node $v \in F(\widehat{\mathcal{G}})$ at least $s+1$ times. (Otherwise, there would exist a graph in $\widehat{\mathcal{G}}$ and a starting node in the main cycle, for which one node would not be explored by U). Hence, for sufficiently large m, the length of D_i is at least $(s+1)|F(\widehat{\mathcal{G}})| \geq \frac{m^2}{5} m^{\frac{(c-1)(1-\epsilon)}{c}}$, because $s \geq \frac{m^2}{5}$. Therefore, the length of U is at least $p\frac{m^2}{5} m^{\frac{(c-1)(1-\epsilon)}{c}} = \frac{1}{5} m^{\epsilon} m^2 m^{\frac{(c-1)(1-\epsilon)}{c}} = \frac{1}{5} m^{2+\epsilon+\frac{(c-1)(1-\epsilon)}{c}} = \frac{1}{5} m^{2+2\epsilon+(\frac{c-1}{c}+\frac{1-2c}{c}\epsilon)}$. Since $\epsilon < \frac{c-1}{2c-1}$, we have $(\frac{c-1}{c} + \frac{1-2c}{c}\epsilon) > 0$. Therefore, the length of U is in $\omega((2m^{1+\epsilon} + m^{\epsilon})^2) = \omega(n^2)$, and hence exploration time is in $\omega(n^2)$. ◀

Since $\epsilon < \frac{1}{2}$ implies $\frac{\epsilon}{1+\epsilon} < \frac{1}{3}$, Theorem 10 yields the following corollary.

▶ **Corollary 11.** *For any $\delta < \frac{1}{3}$, any exploration algorithm using advice of size $o(n^{\delta})$ must take time $\omega(n^2)$ on some n-node graph, for arbitrarily large n.*

3.2 Instance oracle

For the instance oracle we show a general lower bound on the size of advice needed to achieve a given exploration time. The main corollaries of this lower bound are:

- the size of advice $\Theta(n \log n)$ from Proposition 5, sufficient to achieve linear exploration time, cannot be beaten;
- for advice of linear size, exploration time must be quadratic.

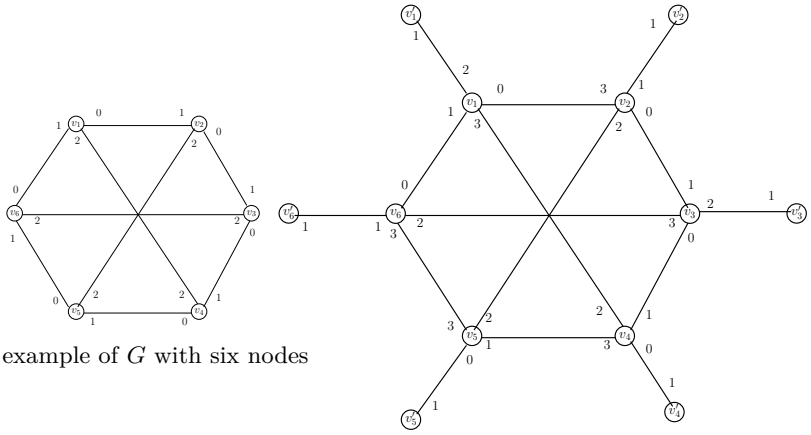
To prove our lower bound we will use the following construction.

Let G be an $\frac{n}{4}$-regular $\frac{n}{2}$-node graph, where n is divisible by 4. We can use, for example the complete bipartite graph with $\frac{n}{2}$ nodes. Let $m = \frac{n}{2}$. Let v_1, v_2, $\cdots$, v_m be the nodes of G. Let $x = (x_1, x_2, \cdots, x_m)$ be a sequence of m integers where $0 \leq x_i \leq \frac{m}{2} - 1$, for $i = 1, \cdots, m$. Let X be the set of all such sequences.

We construct an n-node graph G_x as follows. For each $i = 1, \cdots, m$, add a new node v'_i of degree 1 to G. Replace the port number x_i at v_i by port number $\frac{m}{2}$. Add the edge (v_i, v'_i) with the port number x_i at v_i. An example of the construction of G_x from G is shown in Fig. 2. Let $\mathcal{G}_X$ be the set of all possible graphs G_x constructed from G.

▶ **Theorem 12.** *For any function $\phi : \mathbb{N} \longrightarrow \mathbb{N}$, and for any exploration algorithm using advice of size $o(n\phi(n))$, this algorithm must take time $\Omega(\frac{n^2}{2^{\phi(n)}})$ on some n-node graph from the family $\mathcal{G}_X$, for arbitrarily large integers n.*

Proof. Let n be divisible by 4. We show that if the size of the advice is at most $\frac{n\phi(n)}{4} - 1$, then there exists an n-node graph in the family $\mathcal{G}_X$, for which the time required for exploration is $\Omega(\frac{n^2}{2^{\phi(n)}})$. We have $|\mathcal{G}_X| = |X| = (\frac{m}{2})^m$. There are fewer than $2^{\frac{m\phi(2m)}{2}} = (2^{\phi(2m)})^{\frac{m}{2}}$ different binary strings of length at most $\frac{2m\phi(2m)}{4} - 1 = \frac{n\phi(n)}{4} - 1$. By the pigeonhole principle, there exists a family of graphs $\mathcal{G} \subset \mathcal{G}_X$, of size at least $\frac{(\frac{m}{2})^m}{(2^{\phi(2m)})^{\frac{m}{2}}}$, such that all the graphs in $\mathcal{G}$ get the same advice. Let $Y = \{x \in X : G_x \in \mathcal{G}\}$. Let $z = \frac{m}{2^{\phi(2m)+2}}$ and let



(a) An example of G with six nodes

(b) G_x for the sequence $x = (2, 1, 2, 0, 0, 1)$

Figure 2 The construction of G_x from G.

$J = \{j : |\{x_j : x \in Y\}| \geq z\}$. Intuitively, J is the set of indices, for which the set of terms of sequences x that produce graphs from $\mathcal{G}$ is large. Let $p = |J|$.

Claim: $p > \frac{m}{2}$.

Proof. We prove the claim by contradiction. Suppose that $p \leq \frac{m}{2}$. Since $p \leq \frac{m}{2}$ and $z < \frac{m}{2}$, we have $(\frac{m}{2})^p \cdot z^{m-p} \leq (\frac{m}{2})^{\frac{m}{2}} \cdot z^{\frac{m}{2}}$. Note that for all $i \in \{1, 2, \cdots, m\} \setminus J$, we have $|\{x_i : x \in Y\}| < z$, and for $j \in J$, we have $|\{x_j : x \in Y\}| \leq \frac{m}{2}$. Therefore, $|\mathcal{G}| < (\frac{m}{2})^p \cdot z^{m-p}$. Hence, $|\mathcal{G}| < (\frac{m}{2})^{\frac{m}{2}} (\frac{m}{2^{\phi(2m)+2}})^{\frac{m}{2}} = \frac{m^m}{2^{\frac{m}{2}} (2^{\phi(2m)+2})^{\frac{m}{2}}} < |\mathcal{G}|$, which is a contradiction. This proves the claim. ◀

Consider any exploration algorithm for the class $\mathcal{G}$. There exists a graph $G_x \in \mathcal{G}$, such that, at each node v_j of G_x, for $j \in J$, the agent must take all the ports in $\{x_j : x \in Y\}$. Indeed, suppose that the agent does not take some port x_j, where $j \in J$ and $x \in Y$. Consider the exploration of any graph $G_{x'} \in \mathcal{G}$, where $x'_j = x_j$. Since the agent can visit v'_j only coming from v_j, using port x'_j in $G_{x'}$, the node v'_j remains unexplored, as the port x'_j at v_j is never used, which is a contradiction. Hence, the agent must visit at least $\frac{m}{2^{\phi(2m)+2}}$ ports at each node v_j for $j \in J$. Since $|J| > \frac{m}{2}$, the time required for exploration is at least $\frac{m^2}{2^{\phi(2m)+3}}$, i.e., it is at least $\frac{n^2}{2^{\phi(n)+5}}$. ◀

If $\phi(n) = c$ where c is a constant, then Theorem 12 implies that any exploration algorithm using advice of size at most $\frac{cn}{2}$, must take time at least $\frac{n^2}{2^{c+3}}$. This implies that, if the size of advice is at most $c'n$, for any constant c', then exploration time is $\Omega(n^2)$. Hence we have the following corollary.

▶ **Corollary 13.** *Any exploration algorithm using advice of size $O(n)$ must take time $\Omega(n^2)$ on some n-node graph, for arbitrarily large n.*

For $\phi(n) \in o(\log n)$, Theorem 12 implies an exploration time $\omega(n)$ which shows that the upper bound on the size of advice from Proposition 5 is asymptotically tight for exploration in linear time. The following corollary improves this statement significantly, showing that exploration time is very sensitive to the size of advice at the threshold $\Theta(n \log n)$ of the latter.

▶ **Corollary 14.** *Consider any constant $\epsilon < 2$. Any exploration algorithm using advice of size $o(n \log n)$ must take time $\Omega(n^\epsilon)$, on some n-node graph, for arbitrarily large n.*

Proof. If the size of advice is $o(n \log n)$, then it is $n\phi(n)$, where $\phi(n) = \frac{\log n}{f(n)}$, with $f(n) \to \infty$ as $n \to \infty$. Theorem 12 implies that exploration time must be $\Omega\left(\frac{n^2}{2^{\frac{\log n}{f(n)}}}\right) = \Omega\left(\frac{n^2}{n^{\frac{1}{f(n)}}}\right)$. Since, for any constant $\delta > 0$, we have $n^{\frac{1}{f(n)}} \in O(n^\delta)$, the corollary holds. ◀

4 Exploration of Hamiltonian graphs

In this section we turn attention to Hamiltonian graphs. These graphs have a special feature from the point of view of exploration: with sufficiently large advice of appropriate type, the agent can explore a Hamiltonian graph without any loss of time, visiting each node exactly once, i.e., in time $n-1$, for n-node graphs. Indeed, an instance oracle can give as advice the sequence of port numbers along a Hamiltonian cycle, from the starting node of the agent, and then the agent takes the prescribed ports in $n-1$ consecutive steps. Since it is enough to give $n-1$ port numbers, and the binary representation of each port number uses $O(\log n)$ bits, advice of size $O(n \log n)$, given by an instance oracle, suffices.

We show that neither the quality nor the size of advice can be decreased to achieve the goal of optimal exploration of Hamiltonian graphs. To prove the first statement, we show a graph which is impossible to explore in time $n-1$ when advice of any size is given by a map oracle. Indeed, we construct an n-node Hamiltonian graph for which even knowing the entire map of the graph (but not knowing its starting node) an agent must use time $\Omega(n^2)$ to explore it. To prove the second statement, we construct a class of n-node Hamiltonian graphs for which advice of size $o(n \log n)$, even given by an instance oracle, is not enough to permit exploration of graphs in this class in time $n-1$. Indeed, we show more: any exploration algorithm using such advice must exceed the optimal time $n-1$ by a summand n^ϵ, for any $\epsilon < 1$, on some graph of this class.

In order to prove the first result, we construct a $(3n)$-node Hamiltonian graph $\tilde{G}$ from the n-node graph $\widehat{G}$ described in Section 3.1. First, we consider an $\frac{m}{2}$-regular m-node Hamiltonian graph H (for example, the complete bipartite graph). Let $v_1, v_2, \cdots, v_m$ be the nodes of H along a Hamiltonian cycle. The graph $\widehat{G}$ is constructed from H as described in Section 3.1, where the Hamiltonian path $(v_1, v_2, \cdots, v_m)$ is taken as the spanning tree T. We construct the Hamiltonian graph $\tilde{G}$ from the graph $\widehat{G}$ as follows. Denote by $d(v)$ the degree of node v in $\widehat{G}$. For each node v in $\widehat{G}$, consider a cycle of three nodes $v(1)$, $v(2)$, and $v(3)$, in $\tilde{G}$, with port numbers $3d(v), 3d(v)+1$ in clockwise order at each of these three nodes. For each edge (u, v) in $\widehat{G}$, such that the port numbers corresponding to this edge are p at u and q at v, add, in $\tilde{G}$, the edges $(u(i), v(j))$, for $1 \leq i, j \leq 3$, with the following port numbers. The port numbers corresponding to edge $(u(i), v(j))$ are: $p + (j-1)d(u)$ at $u(i)$ and $q + (i-1)d(v)$ at $v(j)$, see Fig. 3.

▶ **Lemma 15.** *The graph $\tilde{G}$ is Hamiltonian.*

Let $\mathcal{A}$ be an exploration algorithm for $\tilde{G}$ starting from node $y_i(1)$, for some $i \leq m$. We describe the following algorithm $\mathcal{A}^*$ on $\widehat{G}$, starting from node y_i. Ignore all moves of $\mathcal{A}$ taking port $3d(v)$ or $3d(v)+1$ at a node $v(j)$, for $1 \leq j \leq 3$, of $\tilde{G}$. Replace every move of $\mathcal{A}$ taking port $r = p + (i-1)d(v)$, at node $v(j)$, for $1 \leq j \leq 3$, in $\tilde{G}$, where $0 \leq p \leq d(v)-1$, by a move taking port p in $\widehat{G}$.

Then the agent executing $\mathcal{A}^*$ in $\widehat{G}$, starting from the main cycle, explores all the nodes. The time used by $\mathcal{A}^*$ in $\widehat{G}$ does not exceed the time used by $\mathcal{A}$ in $\tilde{G}$. Since, by Theorem 9, any

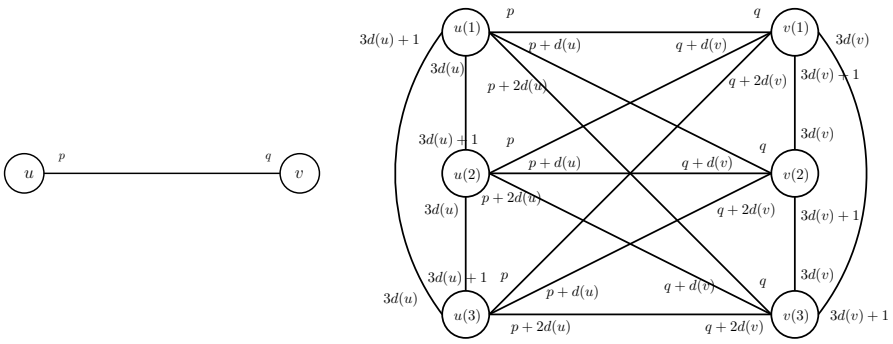


Figure 3 The construction of $\tilde{G}$ from $\widehat{G}$.

exploration algorithm for $\widehat{G}$, starting from the main cycle, must take time $\Omega(n^2)$, algorithm $\mathcal{A}$ must take time $\Omega(n^2)$ to explore $\tilde{G}$. Replacing $3n$ by n we have the following theorem.

▶ **Theorem 16.** *Any exploration algorithm using any advice given by a map oracle must take time $\Omega(n^2)$ on some n-node Hamiltonian graph, for arbitrarily large n.*

Our last result shows that advice of size $o(n \log n)$ causes significant increase of exploration time for some Hamiltonian graphs, as compared to optimal time $n-1$ achievable with advice of size $O(n \log n)$, given by an instance oracle.

▶ **Theorem 17.** *For any constant $\epsilon < 1$, and for any exploration algorithm using advice of size $o(n \log n)$, this algorithm must take time $n + n^{\epsilon}$, on some n-node Hamiltonian graph, for arbitrarily large n.*

5 Conclusion

Most of our lower bounds on the size of advice are either exactly or asymptotically tight. The lower bound $\log \log \log n - \Theta(1)$ on the size of advice sufficient to explore all n-node graphs in polynomial time is exactly tight: with advice of any such size, polynomial exploration is possible, and with advice of any smaller size it is not. For an instance oracle, the lower bound $\Omega(n \log n)$ on the size of advice sufficient to explore n-node graphs in $O(n)$ time is asymptotically tight, as we gave a linear time exploration algorithm using advice of size $O(n \log n)$. An exception to this tightness is the lower bound on the size of advice given by a map oracle, permitting exploration in time $O(n^2)$. While the natural upper bound is $O(n \log n)$, our lower bound is only n^{δ} for any $\delta < \frac{1}{3}$. Hence the main remaining question is:

What is the smallest advice, given by a map oracle, permitting exploration of n-node graphs in time $O(n^2)$?

Acknowledgements. We are grateful to Adrian Kosowski for early discussions on the subject of this paper and for drawing our attention to [10].

References

1 Serge Abiteboul, Stephen Alstrup, Haim Kaplan, Tova Milo, and Theis Rauhe. Compact labeling scheme for ancestor queries. *SIAM J. Comput.*, 35(6):1295–1309, 2006. doi:10.1137/S0097539703437211.

2 Susanne Albers and Monika Rauch Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29(4):1164–1188, 2000. doi:10.1137/S009753979732428X.

3 Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 218–223. IEEE Computer Society, 1979. doi:10.1109/SFCS.1979.34.

4 Baruch Awerbuch, Margrit Betke, Ronald L. Rivest, and Mona Singh. Piecemeal graph exploration by a mobile robot. *Inf. Comput.*, 152(2):155–172, 1999. doi:10.1006/inco.1999.2795.

5 Eldad Bar-Eli, Piotr Berman, Amos Fiat, and Peiyuan Yan. Online navigation in a room. *J. Algorithms*, 17(3):319–341, 1994. doi:10.1006/jagm.1994.1039.

6 Michael A. Bender, Antonio Fernández, Dana Ron, Amit Sahai, and Salil P. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Inf. Comput.*, 176(1):1–21, 2002. doi:10.1006/inco.2001.3081.

7 Michael A. Bender and Donna K. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 75–85. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365703.

8 Margrit Betke, Ronald L. Rivest, and Mona Singh. Piecemeal learning of an unknown environment. *Machine Learning*, 18(2-3):231–254, 1995. doi:10.1007/BF00993411.

9 Avrim Blum, Prabhakar Raghavan, and Baruch Schieber. Navigating in unfamiliar geometric terrain. *SIAM J. Comput.*, 26(1):110–137, 1997. doi:10.1137/S0097539791194931.

10 Allan Borodin, Walter L. Ruzzo, and Martin Tompa. Lower bounds on the length of universal traversal sequences. *J. Comput. Syst. Sci.*, 45(2):180–203, 1992. doi:10.1016/0022-0000(92)90046-L.

11 Jérémie Chalopin, Shantanu Das, and Adrian Kosowski. Constructing a map of an anonymous graph: Applications of universal sequences. In Chenyang Lu, Toshimitsu Masuzawa, and Mohamed Mosbah, editors, *Principles of Distributed Systems - 14th International Conference, OPODIS 2010, Tozeur, Tunisia, December 14-17, 2010. Proceedings*, volume 6490 of *Lecture Notes in Computer Science*, pages 119–134. Springer, 2010. doi:10.1007/978-3-642-17653-1_10.

12 Xiaotie Deng, Tiko Kameda, and Christos H. Papadimitriou. How to learn an unknown environment I: the rectilinear case. *J. ACM*, 45(2):215–245, 1998. doi:10.1145/274787.274788.

13 Dariusz Dereniowski and Andrzej Pelc. Drawing maps with advice. *J. Parallel Distrib. Comput.*, 72(2):132–143, 2012. doi:10.1016/j.jpdc.2011.10.004.

14 Krzysztof Diks, Pierre Fraigniaud, Evangelos Kranakis, and Andrzej Pelc. Tree exploration with little memory. *J. Algorithms*, 51(1):38–63, 2004. doi:10.1016/j.jalgor.2003.10.002.

15 Stefan Dobrev, Rastislav Královic, and Euripides Markou. Online graph exploration with advice. In Guy Even and Magnús M. Halldórsson, editors, *Structural Information and Communication Complexity - 19th International Colloquium, SIROCCO 2012, Reykjavik, Iceland, June 30-July 2, 2012, Revised Selected Papers*, volume 7355 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 2012. doi:10.1007/978-3-642-31104-8_23.

16 Christian A. Duncan, Stephen G. Kobourov, and V. S. Anil Kumar. Optimal constrained graph exploration. *ACM Trans. Algorithms*, 2(3):380–402, 2006. doi:10.1145/1159892.1159897.

17 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theor. Comput. Sci.*, 412(24):2642–2656, 2011. doi:10.1016/j.tcs.2010.08.007.

18 Pierre Fraigniaud, Cyril Gavoille, David Ilcinkas, and Andrzej Pelc. Distributed computing with advice: information sensitivity of graph coloring. *Distributed Computing*, 21(6):395–403, 2009. doi:10.1007/s00446-008-0076-y.

19 Pierre Fraigniaud and David Ilcinkas. Digraphs exploration with little memory. In Volker Diekert and Michel Habib, editors, *STACS 2004, 21st Annual Symposium on Theoretical Aspects of Computer Science, Montpellier, France, March 25-27, 2004, Proceedings*, volume 2996 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2004. doi:10.1007/978-3-540-24749-4_22.

20 Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Tree exploration with advice. *Inf. Comput.*, 206(11):1276–1287, 2008. doi:10.1016/j.ic.2008.07.005.

21 Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Communication algorithms with advice. *J. Comput. Syst. Sci.*, 76(3-4):222–232, 2010. doi:10.1016/j.jcss.2009.07.002.

22 Pierre Fraigniaud, Amos Korman, and Emmanuelle Lebhar. Local MST computation with short advice. *Theory Comput. Syst.*, 47(4):920–933, 2010. doi:10.1007/s00224-010-9280-9.

23 Emanuele G. Fusco and Andrzej Pelc. Trade-offs between the size of advice and broadcasting time in trees. *Algorithmica*, 60(4):719–734, 2011. doi:10.1007/s00453-009-9361-9.

24 Emanuele G. Fusco, Andrzej Pelc, and Rossella Petreschi. Topology recognition with advice. *Inf. Comput.*, 247:254–265, 2016. doi:10.1016/j.ic.2016.01.005.

25 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *J. Algorithms*, 53(1):85–112, 2004. doi:10.1016/j.jalgor.2004.05.002.

26 David Ilcinkas, Dariusz R. Kowalski, and Andrzej Pelc. Fast radio broadcasting with advice. *Theor. Comput. Sci.*, 411(14-15):1544–1557, 2010. doi:10.1016/j.tcs.2010.01.004.

27 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010. doi:10.1007/s00446-010-0095-3.

28 Robert Krauthgamer, editor. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. SIAM, 2016. doi:10.1137/1.9781611974331.

29 Nicolas Nisse and David Soguet. Graph searching with advice. *Theor. Comput. Sci.*, 410(14):1307–1318, 2009. doi:10.1016/j.tcs.2008.08.020.

30 Petrisor Panaite and Andrzej Pelc. Exploring unknown undirected graphs. *J. Algorithms*, 33(2):281–295, 1999. doi:10.1006/jagm.1999.1043.

31 Petrisor Panaite and Andrzej Pelc. Optimal broadcasting in faulty trees. *J. Parallel Distrib. Comput.*, 60(5):566–584, 2000. doi:10.1006/jpdc.2000.1625.

32 Andrzej Pelc and Anas Tiane. Efficient grid exploration with a stationary token. *Int. J. Found. Comput. Sci.*, 25(3):247–262, 2014. doi:10.1142/S0129054114500129.

33 Nageswara S. V. Rao, Srikumar Kareti, Weimin Shi, and S. Sitharama Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms, Jul 1993. doi:10.2172/10180101.

34 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, 2008. doi:10.1145/1391289.1391291.

35 A. Pelc Y. Dieudonné. Impact of knowledge on election time in anonymous networks. In *Proc. 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2017)*, 2017. doi:10.1007/978-3-642-17653-1_10.

Combinatorial Secretary Problems with Ordinal Information*†

Martin Hoefer[1] and Bojana Kodric[2]

1 **Institute of Computer Science, Goethe-University Frankfurt am Main, Frankfurt, Germany**
mhoefer@cs.uni-frankfurt.de

2 **MPI for Informatics and Saarland University, Saarland Informatics Campus, Saarbrücken, Germany**
bojana@mpi-inf.mpg.de

Abstract

The secretary problem is a classic model for online decision making. Recently, combinatorial extensions such as matroid or matching secretary problems have become an important tool to study algorithmic problems in dynamic markets. Here the decision maker must know the numerical value of each arriving element, which can be a demanding informational assumption. In this paper, we initiate the study of combinatorial secretary problems with *ordinal information*, in which the decision maker only needs to be aware of a preference order consistent with the values of arrived elements. The goal is to design online algorithms with small competitive ratios.

For a variety of combinatorial problems, such as bipartite matching, general packing LPs, and independent set with bounded local independence number, we design new algorithms that obtain constant competitive ratios.

For the matroid secretary problem, we observe that many existing algorithms for special matroid structures maintain their competitive ratios even in the ordinal model. In these cases, the restriction to ordinal information does not represent any additional obstacle. Moreover, we show that ordinal variants of the submodular matroid secretary problems can be solved using algorithms for the linear versions by extending [18]. In contrast, we provide a lower bound of $\Omega(\sqrt{n}/(\log n))$ for algorithms that are oblivious to the matroid structure, where n is the total number of elements. This contrasts an upper bound of $O(\log n)$ in the cardinal model, and it shows that the technique of thresholding is not sufficient for good algorithms in the ordinal model.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Secretary Problem, Matroid Secretary, Ordinal Information, Online Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.133

1 Introduction

The secretary problem is a classic approach to model online decision making under uncertain input. The interpretation is that a firm needs to hire a secretary. There are n candidates and they arrive sequentially in random order for an interview. Following an interview, the firm learns the value of the candidate, and it has to make an immediate decision about hiring him

* A full version is available at http://arxiv.org/abs/1702.01290.
† This work was supported by DFG Cluster of Excellence MMCI at Saarland University.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 133; pp. 133:1–133:14


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

before seeing the next candidate(s). If the candidate is hired, the process is over. Otherwise, a rejected candidate cannot be hired at a later point in time. The optimal algorithm is a simple greedy rule that rejects all candidates in an initial learning phase. In the following acceptance phase, it hires the first candidate that is the best among all the ones seen so far. It manages to hire the best candidate with optimal probability $1/e$. Notably, it only needs to know if a candidate is the best seen so far, but no exact numerical values.

Since its introduction [15], the secretary problem has attracted a huge amount of research interest. Recently, a variety of combinatorial extensions have been studied in the literature [7] capturing a variety of fundamental online allocation problems in networks and markets, such as network design [28], resource allocation [25], medium access in networks [22], or competitive admission processes [12]. Prominently, in the matroid secretary problem [8], the elements of a weighted matroid arrive in uniform random order (e.g., weighted edges of an undirected graph G). The goal is to select a max-weight independent set of the matroid (e.g., a max-weight forest of G). The popular *matroid secretary conjecture* claims that for all matroids, there exists an algorithm with a constant *competitive ratio*, i.e., the expected total weight of the solution computed by the algorithm is at least a constant fraction of the total weight of the optimum solution. Despite much progress on special cases, the conjecture remains open. Beyond matroids, online algorithms for a variety of combinatorial secretary problems with downward-closed structure have recently been studied (e.g., matching [28, 25], independent set [22], linear packing problems [26] or submodular versions [18]).

The best known algorithms for matroid or matching secretary problems rely heavily on knowing the exact weight structure of elements. They either compute max-weight solutions to guide the admission process or rely on advanced bucketing techniques to group elements based on their weight. For a decision maker, in many applications it can be quite difficult to determine an exact cardinal preference for each of the incoming candidates. In contrast, in the original problem, the optimal algorithm only needs *ordinal information* about the candidates. This property provides a much more robust guarantee, since the numerical values can be arbitrary, as long as they are consistent with the preference order.

In this paper, we study algorithms for combinatorial secretary problems that rely only on ordinal information. We assume that there is an unknown value for each element, but our algorithms only have access to the *total order* of the elements arrived so far, which is consistent with their values. We term this the *ordinal model*; as opposed to the *cardinal model*, in which the algorithm learns the exact values. We show bounds on the competitive ratio, i.e., we compare the quality of the computed solutions to the optima in terms of the exact underlying but unknown numerical values. Consequently, competitive ratios for our algorithms are robust guarantees against uncertainty in the input. Our approach follows a recent line of research by studying the potential of algorithms with ordinal information to approximate optima based on numerical values [4, 3, 1, 10].

1.1 Our Contribution

We first point out that many algorithms proposed in the literature continue to work in the ordinal model. In particular, a wide variety of algorithms for variants of the matroid secretary problem with constant competitive ratios continue to obtain their guarantees in the ordinal model (see Table 1 for an overview). This shows that many results in the literature are much stronger, since the algorithms require significantly less information. Notably, the algorithm of [9] extends to the ordinal model and gives a ratio of $O(\log^2 r)$ for general matroids, where r is the rank of the matroid. In contrast, the improved algorithms with ratios of $O(\log r)$ and $O(\log \log r)$ [8, 30, 16] are not applicable in the ordinal model.

For several combinatorial secretary problems we obtain new algorithms for the ordinal model. For online bipartite matching we give an algorithm that is $2e$-competitive. More generally, it obtains a ratio of $3e$ when the value of the matching is a submodular set function of the chosen edges. We also extend this result to online packing LPs with at most d non-zero entries per variable. Here we obtain an $O(d^{(B+1)/B})$-competitive algorithm, where B is a tightness parameter of the constraints. Another extension is matching in general graphs, for which we give a 8.78-competitive algorithm.

We give an $O(\alpha_1^2)$-competitive algorithm for the online weighted independent set problem in graphs, where α_1 is the local independence number of the graph. For example, for the prominent case of unit-disk graphs, $\alpha_1 = 5$ and we obtain a constant-competitive algorithm.

For matroids, we extend a result of [18] to the ordinal model: The reduction from submodular to linear matroid secretary can be done with ordinal information for marginal weights of the elements. More specifically, we show that whenever there is an algorithm that solves the matroid secretary problem in the ordinal model on some matroid class and has a competitive ratio of α, there is also an algorithm for the submodular matroid secretary problem in the ordinal model on the same matroid class with a competitive ratio of $O(\alpha^2)$. The ratio can be shown to be better if the linear algorithm satisfies some further properties.

Lastly, we consider the importance of knowing the weights, ordering, and structure of the domain. For algorithms that have complete ordinal information but cannot learn the specific matroid structure, we show a lower bound of $\Omega(\sqrt{n}/(\log n))$, even for partition matroids, where n is the number of elements in the ground set. This bound contrasts the $O(\log^2 r)$-competitive algorithm and indicates that learning the matroid structure is crucial in the ordinal model. Moreover, it contrasts the cardinal model, where thresholding algorithms yield $O(\log r)$-competitive algorithms without learning the matroid structure.

For structural reasons, we present our results in a slightly different order. We first discuss the matroid results in Section 3. Then we proceed with matching and packing in Section 4 and independent set in Section 5. Due to spatial constraints, all missing proofs are deferred to the full version [23].

2 Preliminaries and Related Work

In the typical problem we study, there is a set E of elements arriving sequentially in random order. The algorithm knows $n = |E|$ in advance. It must accept or reject an element before seeing the next element(s). There is a set $\mathcal{S} \subseteq 2^E$ of *feasible solutions*. $\mathcal{S}$ is downward-closed, i.e., if $S \in \mathcal{S}$, then $S' \in \mathcal{S}$ for every $S' \subseteq S$. The goal is to accept a feasible solution that maximizes an objective function f. In the *linear version*, each element has a *value* or *weight* w_e, and $f(S) = \sum_{e \in S} w_e$. In the *submodular version*, f is submodular and $f(\emptyset) = 0$.

In the linear ordinal model, the algorithm only sees a strict total order over the elements seen so far that is consistent with their weights (ties are broken arbitrarily). For the submodular version, we interpret the value of an element as its marginal contribution to a set of elements. In this case, our algorithm has access to an *ordinal oracle* $\mathcal{O}(S)$. For every subset S of arrived elements, $\mathcal{O}(S)$ returns a total order of arrived elements consistent with their marginal values $f(e|S) = f(S \cup \{e\}) - f(S)$. We chose this particular model because it elegantly aligns with the online setting. Both stronger and weaker definitions are possible and these are interesting avenues for future work.

Given this information, we strive to design algorithms with small competitive ratio $f(S^*)/\mathbb{E}[f(S_{alg})]$. Here S^* is an optimal feasible solution and S_{alg} the solution returned by the algorithm. Note that S_{alg} is a random variable due to random-order arrival and possible internal randomization of the algorithm.

Table 1 Existing algorithms for matroid secretary problems that provide the same guarantee in the ordinal model.

Matroid	general	k-uniform	graphic	cographic	transversal	laminar	regular
Ratio	$O(\log^2 r)$	$1 + O(\sqrt{1/k})$, e	$2e$	$3e$	16	$3\sqrt{3}e$	$9e$
Reference	[9]	[15, 27, 6]	[28]	[31]	[13]	[24]	[14]

In the matroid secretary problem, the pair $\mathcal{M} = (E, \mathcal{S})$ is a matroid. We summarize in Table 1 some of the existing results for classes of the (linear) problem that transfer to the ordinal model. The algorithms for all restricted matroid classes other than the graphic matroid assume a-priori complete knowledge of the matroid – only weights are revealed online. The algorithms do not use cardinal information, their decisions are based only on ordinal information. As such, they translate directly to the ordinal model. Notably, the algorithm from [9] solves even the general submodular matroid secretary problem in the ordinal model.

2.1 Related Work

Our work is partly inspired by [4, 5], who study ordinal approximation algorithms for classical optimization problems. They design constant-factor approximation algorithms for matching and clustering problems with ordinal information and extend the results to truthful mechanisms. Our approach here differs due to online arrival. Anshelevich et al. [3] examine the quality of randomized social choice mechanisms when agents have metric preferences but only ordinal information is available to the mechanism. Previously, [1, 10] studied ordinal measures of efficiency in matchings, for instance the average rank of an agent's partner.

The literature on the secretary problem is too broad to survey here. We only discuss directly related work on online algorithms for combinatorial variants. In[29, 21], the authors study hiring a single secretary when only a partial ordering of the candidates is available. For multiple-choice secretary, where we can select any k candidates, there are algorithms with ratios that are constant and asymptotically decreasing in k [27, 6]. More generally, the matroid secretary problem has attracted a large amount of research interest [8, 11, 30, 16], and the best-known algorithm in the cardinal model has ratio $O(\log \log r)$. For results on specific matroid classes, see the overview in Table 1. Extensions to the submodular version are treated in [9, 18].

Another prominent domain is online bipartite matching, in which one side of the graph is known in advance and the other arrives online in random order, each vertex revealing all incident weighted edges when it arrives [28]. In this case, there is an optimal algorithm with ratio e [25]. Moreover, our paper is related to Göbel et al. [22] who study secretary versions of maximum independent set in graphs with bounded inductive independence number ρ. They derive an $O(\rho^2)$-competitive algorithm for unweighted and an $O(\rho^2 \log n)$-competitive algorithm for weighted independent set.

In addition, algorithms have been proposed for further variants of the secretary problem, e.g., the temp secretary problem (candidates hired for a fixed duration) [19], parallel secretary (candidates interviewed in parallel) [17], or local secretary (several firms and limited feedback) [12]. For these variants, some existing algorithms (e.g., for the temp secretary problem in [19]) directly extend to the ordinal model. In general, however, the restriction to ordinal information poses an interesting challenge for future work in these domains.

Algorithm 1: Greedy [18]

Input : ground set E
Output : independent set I

Let $I \leftarrow \emptyset$ and $E' \leftarrow E$.
while $E' \neq \emptyset$ **do**
  Let $u \leftarrow \max_{u'} f(u'|I)$ and $E' \leftarrow E' \setminus \{u\}$;
  if $(I \cup \{u\}$ independent in $\mathcal{M}) \wedge (f(u|I) \geq 0)$ **then** add u to I;

Algorithm 2: Online(p) algorithm [18]

Input : $n = |E|$, size of the ground set
Output : independent set $Q \cap N$

Choose X from the binomial distribution $B(n, 1/2)$.
Reject the first X elements of the input. Let L be the set of these elements.
Let M be the output of Greedy on the set L.
Let $N \leftarrow \emptyset$.
for *each element* $u \in E \setminus L$ **do**
  Let $w(u) \leftarrow 0$.
  if u *accepted by Greedy applied to* $M \cup \{u\}$ **then**
    With probability p do the following:
    Add u to N.
    Let $M_u \subseteq M$ be the solution of Greedy immediately before it adds u to it.
    $w(u) \leftarrow f(u|M_u)$.
  Pass u to Linear with weight $w(u)$.
return $Q \cap N$, where Q is the output of Linear.

3 Matroids

3.1 Submodular Matroids

We start our analysis by showing that – in addition to algorithms for special cases mentioned above – a powerful technique for submodular matroid secretary problems [18] can be adjusted to work even in the ordinal model. More formally, in this section we show that there is a reduction from submodular matroid secretary problems with ordinal information (SMSPO) to linear matroid secretary problems with ordinal information (MSPO). The reduction uses Greedy (Algorithm 1) as a subroutine and interprets the marginal value when added to the greedy solution as the value of an element. These values are then forwarded to whichever algorithm (termed Linear) that solves the linear version of the problem. In the ordinal model, we are unable to see the exact marginal values. Nevertheless, we manage to construct a suitable ordering for the forwarded elements. Consequently, we can apply algorithm Linear as a subroutine to obtain a good solution for the ordinal submodular problem.

Let $\mathcal{M} = (E, \mathcal{S})$ be the matroid, f the submodular function, and E the ground set of elements. The marginal contribution of element u to set M is denoted by $f(u|M) = f(M \cup \{u\}) - f(M)$. Since f can be non-monotone, Greedy in the cardinal model also checks if the marginal value of the currently best element is positive. While we cannot explicitly make this check in the ordinal model, note that $f(u|M) \geq 0 \iff f(M \cup \{u\}) \geq f(M) = f(M \cup \{u'\})$

for every $u' \in M$. Since the ordinal oracle includes the elements of M in the ordering of marginal values, there is a way to check positivity even in the ordinal model. Therefore, our results also apply to non-monotone functions f.

A potential problem with Algorithm 2 is that we must compare marginal contributions of different elements w.r.t. different sets. We can resolve this issue by following the steps of the Greedy subroutine that tries to add new elements to the greedy solution computed on the sample. We use this information to construct a correct ordering over the marginal contributions of elements that we forward to Linear.

▶ **Lemma 1.** *Let us denote by s_u the step of Greedy in which the element u is accepted when applied to $M + u$. Then $s_{u_1} < s_{u_2}$ implies $f(u_1|M_{u_1}) \geq f(u_2|M_{u_2})$.*

Proof. First, note that $M_{u_1} \subset M_{u_2}$ when $s_1 < s_2$. We denote by m_{u_1} the element of M that would be taken in step s_{u_1} if u_1 would not be available. Then we know that $f(u_1|M_{u_1}) \geq f(m_{u_1}|M_{u_1})$. Furthermore, since $s_1 < s_2$, $f(m_{u_1}|M_{u_1}) \geq f(u_2|M_{u_1})$. Lastly, by using submodularity, we know that $f(u_2|M_{u_1}) \geq f(u_2|M_{u_2})$. ◀

When $s_{u_1} = s_{u_2}$, then $M_{u_1} = M_{u_2}$ so the oracle provides the order of marginal values. Otherwise, the lemma yields the ordinal information. Thus, we can construct an ordering for the elements that are forwarded to Linear that is consistent with their marginal values in the cardinal model. Hence, the reduction can be applied in the ordinal model, since the algorithm executes exactly the same as in the cardinal model, and all results from [18] continue to hold. We mention only the main theorem. It implies constant ratios for all problems in Table 1 in the submodular version.

▶ **Theorem 2.** *Given an arbitrary algorithm Linear for MSPO that is α-competitive on a matroid class, there is an algorithm for SMSPO with competitive ratio is at most $24\alpha(3\alpha+1) = O(\alpha^2)$ on the same matroid class. For SMSPO with monotone f, it can be improved to $8\alpha(\alpha+1)$.*

3.2 A Lower Bound

Another powerful technique in the cardinal model is thresholding, where we first sample a constant fraction of the elements to learn their weights. Based on the largest weight observed, we pick a threshold and accept subsequent elements greedily if they exceed the threshold. This approach generalizes the classic algorithm [15] and provides logarithmic ratios for many combinatorial domains [8, 28, 22, 12]. Intuitively, these algorithms *learn the weights but not the structure.*

We show that this technique does not easily generalize to the ordinal model. The algorithms with small ratios in the ordinal model rely heavily on the matroid structure. Indeed, in the ordinal model we show a polynomial lower bound for algorithms in the matroid secretary problem that learn the ordering but not the structure. Formally, we slightly simplify the setting as follows. The algorithm receives the global ordering of all elements in advance. It determines (possibly at random) a threshold position in the ordering. Then elements arrive and are accepted greedily if ranked above the threshold. Note that the algorithm does not use sampling, since in this case the only meaningful purpose of sampling is learning the structure. We call this a *structure-oblivious* algorithm.

▶ **Theorem 3.** *Every structure-oblivious randomized algorithm has a competitive ratio of at least $\Omega(\sqrt{n}/(\log n))$.*

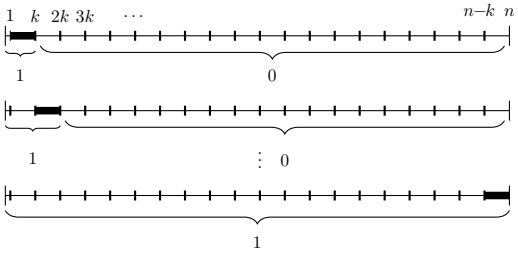


Figure 1 Values for the family of instances described in the proof of Theorem 11, where the position of the "valuable" ones is denoted by the thick segment.

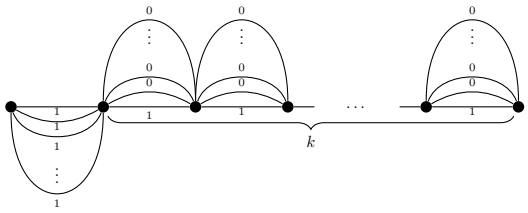


Figure 2 One instance from the family described in the proof of Theorem 11.

Proof. In the proof, we restrict our attention to instances with weights in $\{0, 1\}$ (for a formal justification, see the full version [23]). We give a distribution of such instances on which every deterministic algorithm has a competitive ratio of $\Omega(\sqrt{n}/(\log n))$. Using Yao's principle, this shows the claimed result for randomized algorithms.

All instances in the distribution are based on a graphic matroid (in fact, a partition matroid) of the following form. There is a simple path of $1 + k$ segments. The edges in each segment have weight of 0 or 1. We call the edges with value 1 in the last k segments the "valuable edges". The total number of edges is the same in each instance and equals $n + 1$. All edges in the first segment have value 1 and there is exactly one edge of value 1 in all other segments (that being the aforementioned valuable edges). In the first instance there are in total $k + 1$ edges of value 1 (meaning that there is only one edge in the first segment). In each of the following instances this number is increased by k (in the i-th instance there are $(i - 1) \cdot k + 1$ edges in the first segment) such that the last instance has only edges with value 1 (there are $n - k + 1$ edges in the first segment). The zero edges are always equally distributed on the last k path segments. The valuable edges are lower in the ordering than any non-valuable edge with value 1 (see Figure 1). Each of the instances appears with equal probability of $\frac{k}{n}$ (see Figure 2 for one example instance).

A deterministic algorithm picks a threshold at position i. The expected value of the solution is

$$\mathbb{E}[w(S_{alg})] \leq 1 + \frac{k}{n} \sum_{\ell=1}^{\frac{i}{k}} \frac{k}{\ell} \leq 1 + \frac{k^2}{n} \log \frac{i}{k} \leq \frac{k^2}{n} \log \frac{n}{k} + 1 \ ,$$

where log denotes the natural logarithm and the expression results from observing that the algorithm cannot obtain more than a value of 1 if its threshold i falls above the valuable 1's. Otherwise it gets an additional fraction of k, depending on how close the threshold is positioned to the valuable 1's. For instance, if the threshold is set between 1 and k positions

below the valuable 1's, the algorithm will in expectation select edges of total value of at least $1 + k/2$. This follows from the random arrival order of the edges and the fact that the ratio of valuable to non-valuable edges that the algorithm is ready to accept is at least $1 : 2$. Furthermore, we see that for this distribution of instances the optimal way to set a deterministic threshold is at the lowest position. Using $k = \sqrt{n}$, a lower bound on the competitive ratio is

$$\frac{k}{\frac{k^2}{n}\log\frac{n}{k} + 1} = \frac{n}{k\log\frac{n}{k} + \frac{n}{k}} = \Omega\left(\frac{\sqrt{n}}{\log n}\right) \quad . \qquad \blacktriangleleft$$

4 Matching and Packing

4.1 Bipartite Matching

In this section, we study online bipartite matching. The vertices on the right side of the graph (denoted by R) are static and given in advance. The vertices on the left side (denoted by L) arrive sequentially in a random order. Every edge $e = (r, l) \in R \times L$ has a non-negative weight $w(e) \geq 0$. In the cardinal model, each vertex of L reveals upon arrival the weights of all incident edges. In the ordinal model, we are given a total and strict order of all edges that have arrived so far, consistent with their weights[1]. Before seeing the next vertex of L, the algorithm has to decide to which vertex $r \in R$ (if any) it wants to match the current vertex l. A match that is formed cannot be revoked. The goal is to maximize the total weight of the matching.

The algorithm for the cardinal model in [25] achieves an optimal competitive ratio of e. However, this algorithm heavily exploits cardinal information by repeatedly computing max-weight matchings for the edges seen so far. For the ordinal model, our Algorithm 3 below obtains a competitive ratio of $2e$. While similar in spirit, the main difference is that we rely on a greedy matching algorithm, which is based solely on ordinal information. It deteriorates the ratio only by a factor of 2.

Here we assume to have access to ordinal preferences over all the edges in the graph. Note that the same approach works if the vertices provide correlated (ordinal) preference lists consistent with the edge weights, for every vertex from R and every arrived vertex from L. In this case, the greedy algorithm can still be implemented by iteratively matching and removing a pair that mutually prefers each other the most, and it provides an approximation guarantee of 2 for the max-weight matching (see, e.g., [2]). In contrast, if we receive only preference lists for vertices on one side, there are simple examples that establish super-constant lower bounds on the competitive ratio[2].

▶ **Lemma 4.** *Let the random variable A_v denote the contribution of the vertex $v \in L$ to the output, i.e. weight assigned to v in M. Let $w(M^*)$ denote the value of the maximum-weight matching in G. For $l \in \{\lceil \frac{n}{e} \rceil, \ldots, n\}$,*

$$\mathbb{E}[A_l] \geq \frac{\lceil \frac{n}{e} \rceil}{l-1} \cdot \frac{w(M^*)}{2n} \quad .$$

[1] Ties are broken arbitrarily, but consistently over the arrival process.

[2] Consider a bipartite graph with two nodes on each side (named A,B and 1,2). If we only know that both A and B prefer 1 to 2, the ratio becomes at least 2 even in the offline case. Similar examples imply that the (offline) ratio must grow in the size of the graph.

Algorithm 3: Bipartite Matching

Input : vertex set R and cardinality $n = |L|$
Output : matching M

Let L' be the first $\lfloor \frac{n}{e} \rfloor$ vertices of L, and $M \leftarrow \emptyset$;
for *each* $\ell \in L \setminus L'$ **do**
  $L' \leftarrow L' \cup \{\ell\}$;
  $M^{(\ell)} \leftarrow$ greedy matching on $G[L' \cup R]$;
  Let $e^{(\ell)} \leftarrow (\ell, r)$ be the edge assigned to ℓ in $M^{(\ell)}$;
  if $M \cup \{e^{(\ell)}\}$ is a matching **then** add $e^{(\ell)}$ to M;

Proof. We first show that $e^{(\ell)}$ has a significant expected weight. Then we bound the probability of adding $e^{(\ell)}$ to M.

In step ℓ, $|L'| = \ell$ and the algorithm computes a greedy matching $M^{(\ell)}$ on $G[L' \cup R]$. The current vertex ℓ can be seen as selected uniformly at random from L', and L' can be seen as selected uniformly at random from L. Therefore, $\mathbb{E}[w(M^{(\ell)})] \geq \frac{\ell}{n} \cdot \frac{w(M^*)}{2}$ and $\mathbb{E}[w(e^{(\ell)})] \geq \frac{w(M^*)}{2n}$. Here we use that a greedy matching approximates the optimum by at most a factor of 2 [2].

Edge $e^{(\ell)}$ can be added to M if r has not been matched already. The vertex r can be matched only when it is in $M^{(k)}$ where $\lceil n/e \rceil \leq k \leq l-1$. The probability of r being matched in step k is at most $\frac{1}{k}$ and the order of the vertices in steps $1, \ldots, k-1$ is irrelevant for this event.

$$\Pr[r \text{ unmatched in step } \ell] = \Pr\left[\bigwedge_{k=\lceil n/e \rceil}^{\ell-1} r \notin e^{(k)}\right] \geq \prod_{k=\lceil n/e \rceil}^{\ell-1} \frac{k-1}{k} = \frac{\lceil \frac{n}{e} \rceil - 1}{\ell - 1}$$

We now know that $\Pr[M \cup e^{(\ell)} \text{ is a matching}] \geq \frac{\lfloor n/e \rfloor}{\ell - 1}$. Using this and $\mathbb{E}[w(e^{(\ell)})] \geq \frac{w(M^*)}{2n}$, the lemma follows. ◀

▶ **Theorem 5.** *Algorithm 3 for bipartite matching is* $2e$*-competitive.*

Proof. The weight of matching M can be obtained by summing over random variables A_ℓ.

$$\mathbb{E}[w(M)] = \mathbb{E}\left[\sum_{\ell=1}^{n} A_\ell\right] \geq \sum_{\ell=\lceil n/e \rceil}^{n} \frac{\lfloor n/e \rfloor}{\ell - 1} \cdot \frac{w(M^*)}{2n} = \frac{\lfloor n/e \rfloor}{2n} \sum_{\ell=\lfloor n/e \rfloor}^{n-1} \frac{1}{\ell} \cdot w(M^*)$$

Since $\frac{\lfloor n/e \rfloor}{n} \geq \frac{1}{e} - \frac{1}{n}$ and $\sum_{\ell=\lfloor n/e \rfloor}^{n-1} \frac{1}{l} \geq \ln \frac{n}{\lfloor n/e \rfloor} \geq 1$, it follows that

$$\mathbb{E}[w(M)] \geq \left(\frac{1}{e} - \frac{1}{n}\right) \cdot \frac{w(M^*)}{2} \quad . \qquad \blacktriangleleft$$

We can use the same algorithm and the same analysis in the submodular version, where greedy gives a 3-approximation [20]. It builds the matching by greedily adding an edge that maximizes the marginal improvement of f, which is the information delivered by the ordinal oracle. Hence, our algorithm is $3e$-competitive for the submodular version.

Algorithm 4: Packing LP

Input : capacities $\mathbf{b}$, total number of requests n, probability $p = \frac{e(2d)^{1/B}}{1+e(2d)^{1/B}}$
Output : assignment vector $\mathbf{y}$

Let L' be the first $p \cdot n$ requests, and $\mathbf{y} \leftarrow \mathbf{0}$;
for *each* $j \notin L'$ **do**
  $L' \leftarrow L' \cup \{j\}$;
  $\mathbf{x}^{(L')} \leftarrow$ greedy assignment on the LP for L';
  $\mathbf{y}_j \leftarrow \mathbf{x}_j^{(L')}$;
  if $\neg(\mathbf{A}(\mathbf{y}) \leq \mathbf{b})$ **then** $\mathbf{y}_j \leftarrow \mathbf{0}$;

4.2 Packing

Our results for bipartite matching can be extended to online packing LPs of the form $\max \mathbf{c}^\tau \mathbf{x}$ s.t. $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{0} \leq \mathbf{x} \leq \mathbf{1}$, which model problems with m resources and n online requests coming in random order. Each resource $i \in [m]$ has a capacity b_i that is known in advance, together with the number of requests. Every online request comes with a set of options, where each option has its profit and resource consumption. Once a request arrives, the coefficients of its variables are revealed and the assignment to the variables has to be determined.

Formally, request $j \in [n]$ corresponds to variables $x_{j,1}, \ldots, x_{j,K}$ that represent K options. Each option $k \in [K]$ contributes with profit $c_{j,k} \geq 0$ and has resource consumption $a_{i,j,k} \geq 0$ for resource i. Overall, at most one option can be selected, i.e., there is a constraint $\sum_{k\in[K]} x_{j,k} \leq 1, \forall j \in [n]$. The objective is to maximize total profit while respecting the resource capacities. The offline problem is captured by the following linear program:

$$\max \sum_{j\in[n]} \sum_{k\in[K]} c_{j,k} x_{j,k} \quad \text{s.t.} \quad \sum_{j\in[n]} \sum_{k\in[K]} a_{i,j,k} x_{j,k} \leq b_i \qquad i \in [m]$$
$$\sum_{k\in[K]} x_{j,k} \leq 1 \qquad j \in [n]$$

As a parameter, we denote by d the maximum number of non-zero entries in any column of the constraint matrix $\mathbf{A}$, for which by definition $d \leq m$. We compare the solution to the fractional optimum, which we denote by $\mathbf{x}^*$. The competitive ratio will be expressed in terms of d and the capacity ratio $B = \min_{i\in[m]} \left\lfloor \frac{b_i}{\max_{j\in[n],k\in[K]} a_{i,j,k}} \right\rfloor$.

Kesselheim et al. [25] propose an algorithm that heavily exploits cardinal information – it repeatedly solves an LP-relaxation and uses the solution as a probability distribution over the options. Instead, our Algorithm 4 for the ordinal model is based on greedy assignments in terms of profits $c_{j,k}$. More specifically, the greedy assignment considers variables $x_{j,k}$ in non-increasing order of $c_{j,k}$. It sets a variable to 1 if this does not violate the capacity constraints, and to 0 otherwise.

▶ **Theorem 6.** *Algorithm 4 for online packing LPs is $O(d^{(B+1)/B})$-competitive.*

4.3 Matching in General Graphs

Here we study the case when vertices of a general undirected graph arrive in random order. In the beginning, we only know the number n of vertices. Each edge in the graph has a non-negative weight $w(e) \geq 0$. Each vertex reveals the incident edges to previously arrived

Algorithm 5: General Matching

Input : vertex set V and cardinality $n = |V|$
Output : matching M

Let R be the first $\lfloor \frac{n}{2} \rfloor$ vertices of V;
Let L' be the further $\lfloor \frac{n}{2e} \rfloor$ vertices of V, and $M \leftarrow \emptyset$;
for *each* $\ell \in V \setminus L'$ **do**
  $L' \leftarrow L' \cup \{\ell\}$;
  $M^{(\ell)} \leftarrow$ greedy matching on $G[L' \cup R]$;
  Let $e^{(\ell)} \leftarrow (\ell, r)$ be the edge assigned to ℓ in $M^{(\ell)}$;
  if $M \cup \{e^{(\ell)}\}$ is a matching **then** add $e^{(\ell)}$ to M;

vertices and their weights (cardinal model), or we receive a total order over all edges among arrived vertices that is consistent with the weights (ordinal model). An edge can be added to the matching only in the round in which it is revealed. The goal is to construct a matching with maximum weight.

We can tackle this problem by prolonging the sampling phase and dividing the vertices into "left" and "right" vertices. Algorithm 5 first samples $n/2$ vertices. These are assigned to be the set R, corresponding to the static side of the graph in bipartite matching. The remaining vertices are assigned to be the set L. The algorithm then proceeds by sampling a fraction of the vertices of L, forming a set L'. The remaining steps are exactly the same as in Algorithm 3.

▶ **Theorem 7.** *Algorithm 5 for matching in general graphs is* $12e/(e+1)$*-competitive, where* $12e/(e+1) < 8.78$.

5 Independent Set and Local Independence

In this section, we study maximum independent set in graphs with bounded local independence number. The set of elements are the vertices V of an underlying undirected graph G. Each vertex has a weight $w_v \geq 0$. We denote by $N(v)$ the set of direct neighbors of vertex v. Vertices arrive sequentially in random order and reveal their position in the order of weights of vertices seen so far. The goal is to construct an independent set of G with maximum weight. The exact structure of G is unknown, but we know that G has a bounded local independence number α_1.

▶ **Definition 8.** An undirected graph G has local independence number α_1 if for each node v, the cardinality of every independent set in the neighborhood $N(v)$ is at most α_1.

We propose Algorithm 6, which is inspired by the Sample-and-Price algorithm for matching in [28]. Note that Göbel et al. [22] construct a more general approach for graphs with bounded *inductive* independence number ρ. However, they only obtain a ratio of $O(\rho^2 \log n)$ for the weighted version, where a competitive ratio of $\Omega(\log n / \log^2 \log n)$ cannot be avoided, even in instances with constant ρ. These algorithms rely on ρ-approximation algorithms for the offline problem that crucially exploit cardinal information.

Similar to [28], we reformulate Algorithm 6 into an equivalent approach (Algorithm 7) for the sake of analysis. Given the same arrival order, the same vertices are in the sample. Algorithm 7 drops all vertices from S that have neighbors in S while Algorithm 6 keeps one of them. Hence, $\mathbb{E}[w(S_{Alg_6})] \geq \mathbb{E}[w(S_{Sim})]$. In what follows, we analyze the performance

Algorithm 6: Independent Set in Graphs with Bounded Local Independence Number

Input : $n = |G|$, $p = \sqrt{\alpha_1/(\alpha_1+1)}$
Output : independent set of vertices S

Set $k \leftarrow \text{Binom}(n,p)$, $S \leftarrow \emptyset$;
Reject first k vertices of G, denote this set by G';
Build a maximal independent set of vertices from G' greedily, denote this set by M_1;
for *each* $v \in G \setminus G'$ **do**
    $w^* \leftarrow \max\{w \mid \mathcal{N}(v) \cap M_1\}$;
    if $(v > w^*) \wedge (S \cup \{v\}$ independent set$)$ **then** add v to S;

Algorithm 7: Simulate

Input : $n = |G|$, $p = \sqrt{\alpha_1/(\alpha_1+1)}$
Output : independent set of vertices S

Sort all vertices in G in non-increasing order of value;
Initialize $M_1, M_2 \leftarrow \emptyset$;
for *each* $v \in G$ *in sorted order* **do**
    if $M_1 \cup \{v\}$ *independent set* **then**
        flip a coin with probability p of heads;
        if heads **then** $M_1 \leftarrow M_1 \cup \{v\}$; **else** $M_2 \leftarrow M_2 \cup \{v\}$;
$S \leftarrow M_2$;
for *each* $w \in S$ **do**
    if w has neighbors in S **then** remove w and all his neighbors from S;

of Simulate. The first lemma follows directly from the definition of the local independence number.

▶ **Lemma 9.** $\mathbb{E}[w(M_1)] \geq p \cdot \frac{w(S^*)}{\alpha_1}$, *where* $\alpha_1 \geq 1$ *is the local independence number of* G.

▶ **Lemma 10.** $\mathbb{E}\big[|\mathcal{N}(v) \cap M_2| \;\big|\; v \in M_2\big] \leq \frac{\alpha_1(1-p)}{p}$.

Proof. Let us denote by X_u^1 and X_u^2 the indicator variables for the events $u \in M_1$ and $u \in M_2$ respectively. Then,

$$\begin{aligned}\mathbb{E}\big[|\mathcal{N}(v) \cap M_2| \;\big|\; v \in M_2\big] &= \mathbb{E}\left[\sum_{u \in \mathcal{N}(v)} X_u^2 \;\middle|\; v \in M_2\right] = \sum_{u \in \mathcal{N}(v)} \mathbb{E}\big[X_u^2 \;\big|\; v \in M_2\big] \\ &= \frac{1-p}{p} \sum_{u \in \mathcal{N}(v)} \mathbb{E}\big[X_u^1 \;\big|\; v \in M_2\big] \leq \frac{1-p}{p} \cdot \alpha_1 \ . \end{aligned}$$

◀

▶ **Theorem 11.** *Algorithm 7 for weighted independent set is* $O(\alpha_1^2)$*-competitive, where* α_1 *is the local independence number of the graph.*

Proof. By using Markov's inequality and Lemma 10,

$$\begin{aligned} \Pr[|\mathcal{N}(v) \cap M_2| \geq 1 \;\big|\; v \in M_2] &\leq \alpha_1 \cdot (1-p)/p \\ \text{and} \quad \Pr[|\mathcal{N}(v) \cap M_2| < 1 \;\big|\; v \in M_2] &> 1 - (\alpha_1(1-p)/p) \ . \end{aligned}$$

Thus, we can conclude that

$$\mathbb{E}[w(S)] \geq \left(1 - \alpha_1 \cdot \frac{1-p}{p}\right) \cdot \mathbb{E}[w(M_2)] \geq \left(1 - \alpha_1 \cdot \frac{1-p}{p}\right) \cdot \frac{1-p}{\alpha_1} \cdot w(S^*) \ .$$

The ratio is optimized for $p = \sqrt{\frac{\alpha_1}{\alpha_1+1}}$, which proves the theorem. ◀

As a prominent example, $\alpha_1 = 5$ in the popular class of unit-disk graphs. In such graphs, our algorithm yields a constant competitive ratio for online independent set in the ordinal model.

References

1. David Abraham, Robert Irving, Telikepalli Kavitha, and Kurt Mehlhorn. Popular matchings. *SIAM J. Comput.*, 37(4):1030–1045, 2007.
2. Elliot Anshelevich and Martin Hoefer. Contribution games in networks. *Algorithmica*, 63(1–2):51–90, 2012.
3. Elliot Anshelevich and John Postl. Randomized social choice functions under metric preferences. In *Proc. 25th Intl. Joint Conf. Artif. Intell. (IJCAI)*, pages 46–59, 2016.
4. Elliot Anshelevich and Shreyas Sekar. Blind, greedy, and random: Algorithms for matching and clustering using only ordinal information. In *Proc. 13th Conf. Artificial Intelligence (AAAI)*, pages 390–396, 2016.
5. Elliot Anshelevich and Shreyas Sekar. Truthful mechanisms for matching and clustering in an ordinal world. In *Proc. 12th Conf. Web and Internet Economics (WINE)*, pages 265–278, 2016.
6. Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *Proc. 10th Workshop Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 16–28, 2007.
7. Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exchanges*, 7(2), 2008.
8. Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proc. 18th Symp. Discrete Algorithms (SODA)*, pages 434–443, 2007.
9. MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Trans. Algorithms*, 9(4):32, 2013.
10. Deeparnab Chakrabarty and Chaitanya Swamy. Welfare maximization and truthfulness in mechanism design with ordinal preferences. In *Proc. 5th Symp. Innovations in Theoret. Computer Science (ITCS)*, pages 105–120, 2014.
11. Sourav Chakraborty and Oded Lachish. Improved competitive ratio for the matroid secretary problem. In *Proc. 23rd Symp. Discrete Algorithms (SODA)*, pages 1702–1712, 2012.
12. Ning Chen, Martin Hoefer, Marvin Künnemann, Chengyu Lin, and Peihan Miao. Secretary markets with local information. In *Proc. 42nd Intl. Coll. Automata, Languages and Programming (ICALP)*, volume 2, pages 552–563, 2015.
13. Nedialko Dimitrov and Greg Plaxton. Competitive weighted matching in transversal matroids. *Algorithmica*, 62(1-2):333–348, 2012.
14. Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. *SIAM J. Comput.*, 43(5):1807–1830, 2014.
15. Eugene Dynkin. The optimum choice of the instant for stopping a Markov process. In *Sov. Math. Dokl*, volume 4, pages 627–629, 1963.
16. Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple $O(\log \log(\text{rank}))$-competitive algorithm for the matroid secretary problem. In *Proc. 26th Symp. Discrete Algorithms (SODA)*, pages 1189–1201, 2015.

17 Moran Feldman and Moshe Tennenholtz. Interviewing secretaries in parallel. In *Proc. 13th Conf. Electronic Commerce (EC)*, pages 550–567, 2012.

18 Moran Feldman and Rico Zenklusen. The submodular secretary problem goes linear. In *Proc. 56th Symp. Foundations of Computer Science (FOCS)*, pages 486–505, 2015.

19 Amos Fiat, Ilia Gorelik, Haim Kaplan, and Slava Novgorodov. The temp secretary problem. In *Proc. 23rd European Symp. Algorithms (ESA)*, pages 631–642, 2015.

20 Marshall Fisher, George Nemhauser, and Laurence Wolsey. An analysis of approximations for maximizing submodular set functions-II. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

21 Byrn Garrod and Robert Morris. The secretary problem on an unknown poset. *Random Struct. Algorithms*, 43(4), 2013.

22 Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets and periods. In *Proc. 41st Intl. Coll. Automata, Languages and Programming (ICALP)*, volume 2, pages 508–519, 2014.

23 Martin Hoefer and Bojana Kodric. Combinatorial secretary problems with ordinal information. *CoRR*, abs/1702.01290, 2017. URL: `http://arxiv.org/abs/1702.01290`.

24 Patrick Jaillet, José Soto, and Rico Zenklusen. Advances on matroid secretary problems: Free order model and laminar case. In *Proc. 16th Intl. Conf. Integer Programming and Combinatorial Optimization (IPCO)*, pages 254–265, 2013.

25 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *Proc. 21st European Symp. Algorithms (ESA)*, pages 589–600, 2013.

26 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal beats dual on online packing LPs in the random-order model. In *Proc. 46th Symp. Theory of Computing (STOC)*, pages 303–312, 2014.

27 Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proc. 16th Symp. Discrete Algorithms (SODA)*, pages 630–631, 2005.

28 Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *Proc. 36th Intl. Coll. Automata, Languages and Programming (ICALP)*, pages 508–520, 2009.

29 Ravi Kumar, Silvio Lattanzi, Sergei Vassilvitskii, and Andrea Vattani. Hiring a secretary from a poset. In *Proc. 12th Conf. Economics and Computation (EC)*, 2011.

30 Oded Lachish. O(log log rank) competitive ratio for the matroid secretary problem. In *Proc. 55th Symp. Foundations of Computer Science (FOCS)*, pages 326–335, 2014.

31 José Soto. Matroid secretary problem in the random-assignment model. *SIAM J. Comput.*, 42(1):178–211, 2013.

Selling Complementary Goods: Dynamics, Efficiency and Revenue*

Moshe Babaioff[1], Liad Blumrosen[2], and Noam Nisan[3]

1 **Microsoft Research, Herzlia, Israel**
moshe@microsoft.com
2 **School of Business Administration, The Hebrew University, Jerusalem, Israel**
blumrosen@gmail.com
3 **School of Engineering and Computer Science, The Hebrew University, Jerusalem, Israel; and**
Microsoft Research
noam@cs.huji.ac.il

Abstract

We consider a price competition between two sellers of perfect-complement goods. Each seller posts a price for the good it sells, but the demand is determined according to the sum of prices. This is a classic model by Cournot (1838), who showed that in this setting a monopoly that sells both goods is better for the society than two competing sellers.

We show that non-trivial pure Nash equilibria always exist in this game. We also quantify Cournot's observation with respect to both the optimal welfare and the monopoly revenue. We then prove a series of mostly negative results regarding the convergence of best response dynamics to equilibria in such games.

1998 ACM Subject Classification J.4. Economics, G.2.2 Network Problems

Keywords and phrases Complements, Pricing, Networks, Game Theory, Price of Stability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.134

1 Introduction

In this paper we study a model of a pricing game between two firms that sell goods that are perfect complements to each other. These goods are only demanded in bundles, at equal quantities, and there is no demand for each good by itself. The two sellers simultaneously choose prices p_1, p_2 and the demand at these prices is given by $\mathcal{D}(p_1 + p_2)$ where $\mathcal{D}$ is the demand for the bundle of these two complementary goods. The revenue of seller i is thus $p_i \cdot \mathcal{D}(p_1 + p_2)$, and as we assume zero production costs, this is taken as his utility.

This model was first studied in Cournot's famous work [9]. In [9], Cournot studied two seminal oligopoly models. The first, and the more famous, model is the well known Cournot oligopoly model about sellers who compete through quantities. We study a second model that was proposed by Cournot in the same work, regarding price competition between sellers of perfect complements.[1] Cournot considered a model of a duopoly selling perfect complements, and he suggested zinc and copper as an example. In Cournot's example, a manufacturer of zinc may observe that some of her major customers produce brass (made of zinc and copper);

* The full version of this paper can be found in an arXiv paper under the same title, see https://arxiv.org/abs/1706.00219.

[1] [25] showed that these two different models by Cournot actually share the same formal structure.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 134; pp. 134:1–134:14

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

Therefore, zinc manufacturers indirectly compete with manufacturers of copper, as both target the money of brass producers. Another classic example of a duopoly selling perfect complements is by [12], who studied how owners of two consecutive segments of a canal determine the tolls for shippers; Clearly, every shipper must purchase a permit from both owners for being granted the right to cross the canal. Another, more contemporary, example for perfect complements might be high-tech or pharmaceutical firms that must buy the rights to use two registered patents to manufacture its product; The owners of the two patents quote prices for the usage rights, and these patents can be viewed as perfect complements.

Cournot, in his 1838 book, proved a counterintuitive result saying that competition among multiple sellers of complement goods lead to a *worse* social outcome than the result reached by a monopoly that controls the two sellers. Moreover, both the profits of the firms and the consumer surplus increase in the monopoly outcome. In the legal literature, this phenomenon was termed "the tragedy of the anticommons" (see, [6, 17, 21]). In our work, we will quantify the severity of this phenomenon.

Clearly, if the demand at a sufficiently high price is zero, then there are *trivial* equilibria in which both sellers price prohibitively high, and nothing is sold. This raises the following question: *Do non-trivial equilibria, in which some pairs of items are sold, always exist?* We study this question as well as some natural follow-ups: *What are the revenue and welfare properties of such equilibria? What are the properties of equilibria that might arise as a result of best-response dynamics?*

For the sake of quantification, we study a *discretized version* of this game in which the demand changes only finitely many times. The number of discrete steps in the demand function, also viewed as the number of possible types of buyers, is denoted by n and is called the number of *demand levels*.

Our first result proves the existence of non-trivial pure Nash equilibria.

▶ **Theorem 1.** *For any demand function with n demand levels there exists at least one non-trivial pure Nash equilibrium.*

We prove the theorem using an artificial dynamics which starts from zero prices and continues in steps. In each step, one seller best responds to the other seller's price, and after each seller best responds, the total price of both is symmetrized: both prices are replaced by their average. We show that the total price is monotonically non-decreasing, and thus it terminates after at most n steps in the non-trivial equilibrium of highest revenue and welfare.

In our model, it is easy to observe that there are multiple equilibria for some demand functions. How different can the welfare and revenue of these equilibria be? A useful parameter for bounding the difference, as well as bounding the inefficiency of equilibria, turns out to be D, the ratio between the demand at price 0 and the demand at the highest price v_{max} for which there is non-zero demand.

Consider the following example with two ($n = 2$) types of buyers: a single buyer that is willing to pay "a lot", 2, for the bundle of the two goods, and many, $D - 1 >> 2$, buyers that are willing to pay "a little", 1, each, for the bundle. A monopolist (that controls both sellers) would have sold the bundle at the low price 1. At this price, all the D buyers decide to buy, leading to revenue D and optimal social welfare of $D + 1$. Equilibria here belong to two types: the "bad" equilibria[2] have high prices, $p_1 + p_2 = 2$, (which certainly is an equilibrium when, say, $p_1 = p_2 = 1$) and achieve low revenue and low social welfare of 2. The "good" equilibria

[2] It turns out that in our model there is no conflict between welfare and revenue in equilibria - the lower the total price, the higher the welfare and the revenue in equilibria (see Proposition 6).

have low prices, $p_1 + p_2 = 1$ (which is an equilibrium as long as $p_1, p_2 \geq 1/D$), and achieve optimal social welfare as well as the monopolist revenue, both values are at least D. Thus, we see that the ratio of welfare (and revenue) between the "good" and "bad" equilibria can be very high, as high as $\Omega(D)$. This can be viewed as a negative "Price of Anarchy" result.

We next focus on the best equilibria and present bounds on the "Price of Stability" of this game; We show that the ratio between the optimal social welfare and the best equilibrium revenue[3] is bounded by $O(\sqrt{D})$, and that this is tight when $D = n$. When n is very small, the ratio can only grow as 2^n and not more. In particular, for constant n the ratio is a constant, in contrast to the lower bound of $\Omega(D)$ on "Price of Anarchy" for $n = 2$, presented above.

▶ **Theorem 2.** *For any instance, the optimal welfare and the monopolist revenue are at most* $O(\min\{2^n, \sqrt{D}\})$ *times the revenue of the best equilibrium. These bounds are tight.*

We now turn to discuss how such markets converge to equilibria, and in case of multiple equilibria, which of them will be reached? We consider best response dynamics in which players start with some initial prices and repeatedly best-reply to each other. We study the quality of equilibria reached by the dynamics, compared to the best equilibria.

Clearly, if the dynamics happen to start at an equilibrium, best replying will leave the prices there, whether the equilibrium is good or bad. But what happens in general: which equilibrium will they "converge" to when starting from "natural" starting points, if any, and how long can that take? Zero prices (or other, very low prices) are probably the most natural starting point. However, as can be seen by the example above, starting from zero prices may result in the worst equilibrium.[4] Another natural starting point is a situation where the two sellers form a cartel and decide to post prices that sum to the monopoly price. Indeed, in our example above, if the two sellers equally split the monopoly price, this will be the best equilibrium. However, we know that cartel solutions are typically unstable, and the participants will have incentives to deviate to other prices and thus start a price updating process. We prove a negative result in this context, showing that starting from any split of the monopoly price might result in bad equilibria. We also check what would be the result of dynamics that start at random prices. Again, we prove a negative result showing situations where dynamics starting from random prices almost surely converge to bad equilibria. Finally, we show that convergence might take a long time, even with only two demand levels. Following is a more formal description of these results about the best-response dynamics:

▶ **Theorem 3.** *The following statements hold:*

- *There are instances with* 3 *demand levels for which a best-response dynamics starting from any split of a monopoly price reaches the worst equilibrium that is factor* $\Omega(\sqrt{D})$ *worse than the best equilibrium in terms of both revenue and welfare.*
- *For any* $\epsilon > 0$ *and* $D > 2/\epsilon$ *there are instances with* 2 *demand levels for which a best-response dynamics starting from uniform random prices in* $[0, v_{max}]^2$ *reaches the worst equilibrium with probability* $1 - \epsilon$*, while the best equilibrium has welfare and revenue that is factor* $\epsilon \cdot D$ *larger.*

[3] Note that this also shows the same bounds on the ratio between the optimal welfare and the welfare in the best equilibrium, as well as the ratio between the monopolist revenue and the revenue in the best equilibrium.

[4] In this example, the best response to price of 0 is price of 1. Next, the first seller will move from price of 0 to price of 1 as well, resulting in the worst equilibrium.

- *For any $n \geq 2$ and $\epsilon > 0$ there are instances with n demand levels for which a best-response dynamics starting from uniform random prices in $[0, v_{max}]^2$ almost surely (with probability 1) reaches the worst equilibrium, while the best equilibrium has welfare and revenue that is factor $\Omega(2^n)$ larger.*
- *(Slow convergence.) For any $K > 0$ there is an instance with only 2 demand levels ($n = 2$) and $D < 2$ for which a best-response dynamics continues for at least K steps before converging to an equilibrium.*

Thus, best-reply dynamics may take a very long time to converge, and then typically end up at a very bad equilibrium. While for very simple ($n = 2$) markets we know that convergence will always occur, we do not know whether convergence is assured for every market.

Open Problem. *Do best reply dynamics always converge to an equilibrium or may they loop infinitely?* We do not know the answer even for $n = 3$.

More related work. While this paper studies price competition between sellers of perfect complements, the classic Bertrand competition [5] studied a similar situation between sellers of perfect substitutes. Bertrand competition leads to an efficient outcome with zero profits for the sellers. [4] studied Bertrand-like competition over a network of sellers. In another paper [3], we studied a network of sellers of perfect complements, where we showed how equilibrium properties depend on the graph structure, and we proved price-of-stability results for lines, cycles, trees etc. Chawla and Roughgarden [8] studied the price of anarchy in two-sided markets with consumers interested in buying flows in a graph from multiple sellers, each selling limited bandwidth on a single edge. Their model is fundamentally different than ours (e.g., they consider combinatorial demand by buyers, and sellers with limited capacities) and their PoA results are with respect to unrestricted Nash Equilibrium, while we focus on non-trivial ones (in our model the analysis of PoA is straightforward for unrestricted NE). A similar model was also studied in [7].

[11] extended the complements model of Cournot to accommodate multiple brands of compatible goods. [10] studied pricing strategies for complementary software products. The paper by [14] directly studied the Cournot/Ellet model, but when buyers approach the sellers (or the tollbooths on the canal) sequentially.

[15] discussed best-response dynamics in a Cournot Oligopoly model with linear demand functions, and proved that they converge to equilibria. Another recent paper [19] studied how no-regret strategies converge to Nash equilibria in Cournot and Bertrand oligopoly settings; The main results in [19] are positive, showing how such strategies lead to a positive-payoff outcomes in Bertrand competition, but they do not consider such a model with complement items.

Best-response dynamics is a natural description of how decentralized markets converge to equilibria, see, e.g., [13, 20], or to approximate equilibria, e.g., [2, 24]. The inefficiency of equilibria in various settings has been extensively studied, see, [18, 22, 23, 1, 16].

We continue as follows: Section 2 defines our model and some basic equilibrium properties. In Section 3 we prove the existence of non trivial equilibria. In Section 4 we study the results of best-response dynamics. Finally, Section 5 compares the quality of the best equilibria to the optimal outcomes.

2 Model and Preliminaries

We consider two sellers, each selling a single, homogeneous, divisible good. The sellers have zero manufacturing cost for the good they sell, and an unlimited supply is available from each good. All the buyers in the economy are interested in bundles of these two goods, and the goods are perfect complements for the buyers. That is, each buyer only demands a bundle consists of two goods, in equal quantities[5], and there is no demand for each good separately. The demand for the bundle of the two goods is given by a demand function $\mathcal{D}(\cdot)$, where $\mathcal{D}(p) \in \mathbb{R}_+$ is the quantity of each of the two goods which is demanded when the price for one unit of the bundle of the two goods is $p \in \mathbb{R}_+$.

The sellers simultaneously offer prices for the goods they sell. Each seller offers a single price, and cannot discriminate between buyers. If the two prices offered by the sellers are p and q then $p+q$ is the *total price* and the demand in this market is $\mathcal{D}(p+q)$. The revenue of the seller that posts a price p is thus $p \cdot \mathcal{D}(p+q)$, the revenue of the second seller is $q \cdot \mathcal{D}(p+q)$ and the total revenue of the two selling firms is denoted by $R(p+q) = (p+q) \cdot \mathcal{D}(p+q)$. The maximal revenue that a monopoly that owns the two sellers can achieve is $\sup_x x \cdot \mathcal{D}(x)$ and we use p^* to denote a monopolist price.[6]

Discrete Demand Levels. In this paper we consider discrete demand curves, where potential buyers only have $n \geq 2$ different values denoted by $\vec{v}$, such that $v_1 > v_2 > \cdots > v_n > 0$. The demand at each price v_i is denoted by $d_i = \mathcal{D}(v_i)$, and assuming a downward sloping demand curve we get that $\vec{d}$ is increasing, that is, $0 < d_1 < d_2 < \cdots < d_n$. For convenience, we define $v_0 = \infty$ and $d_0 = 0$. The parameter n is central in our analysis and it denotes the number of *demand levels* in the economy. Another parameter that we frequently use is the *total demand* D, which is the ratio between the highest and lowest demand at non-zero prices, that is $D = d_n/d_1$. In other words, D is the maximal demand d_n measured in units of the minimal non-zero demand d_1 (Note that $D > 1$). The *social welfare* in the economy is the total value generated for the consumers. The social welfare, given a total price x, is $SW(x) = \sum_{i|x<v_i} v_i(d_i - d_{i-1})$, and the optimal welfare is $SW(0) = \sum_{i=1}^{n} v_i(d_i - d_{i-1})$.

Strategies and Equilibria. The sellers engage in a price competition. We say that p is a *best response* to a price q of the other seller if $p \in argmax_{p'}\ p' \cdot \mathcal{D}(p'+q)$, and let the set of all best responses to q be $BR(q)$. We consider the pure Nash equilibria (NE) of this full-information pricing game. A pure Nash equilibrium is a pair of prices such that each price is a best response to the other price, that is, (p, q) such that $p \in BR(q)$ and $q \in BR(p)$.

It is easy to see that NE always exist in this game, but unfortunately some of them are trivial and no item is sold, and thus their welfare is zero; For example, (∞, ∞) is always an equilibrium with zero welfare and revenue. We will therefore focus on a subset of NE that are *non-trivial*, i.e., where some quantity is sold. It is not immediate to see that non-trivial equilibria exist, and we will begin by proving (in Section 3) that such equilibria indeed always exist. On the other hand, we will show that multiplicity of equilibria is a problem even for

[5] This actually assumes that the ratio of demand of the two goods is fixed, as we can normalized the units to assume that it is 1 for both.

[6] Our paper considers demand functions for which the monopoly revenue is attained and a monopolist price exists. When there is more than one price that maximizes the monopoly profit, our claims regarding p^* will hold for each one of these prices. When necessary, we will treat the different prices separately.

this restricted set of equilibria, as there might be an extreme variance in their revenue and efficiency.

2.1 Basic Equilibrium Properties

We now describe some basic structural properties of equilibria in the pricing game between sellers of complement goods. We use these properties throughout the paper.

We start with a simple observation claiming that all best response dynamics lead to a total price which is exactly one of the demand values. This holds as otherwise any seller can slightly increase his price, selling the same quantity and increasing his revenue.

▶ **Observation 4.** *Let $x \leq v_1$ be some price offered by one seller, and $BR(x)$ be a best response of the other seller to the price x. Then, it holds that $x + BR(x) = v_i$ for some $i \in \{1, ..., n\}$. In particular, for every pure non-trivial NE (p, q), it holds that $p + q = v_i$ for some i.*

Next, we prove a useful lemma claiming that the set of equilibria with a particular total price is convex. Intuitively, the idea in the proof is that a seller with a higher offer cares more about changes in the demand than a seller with a lower offer. Therefore, if the seller with the higher offer decided not to deviate to an increased price, clearly the other seller would not deviate as well. The proof of the lemma appears in the full version of the paper.

▶ **Lemma 5.** *If (p, q) is a pure NE then $(x, p + q - x)$ is also a pure NE for every $x \in [\min\{p, q\}, \max\{p, q\}]$. In particular, $((p + q)/2, (p + q)/2)$ is also a pure NE.*

We next observe that there is no conflict between welfare and revenue in equilibrium: an equilibrium with the highest welfare also has the highest equilibrium revenue. This holds since equilibria with lower total price obtain higher revenue and welfare. We can thus say that any equilibrium with minimal total price is the "best" as it is as good as possible on both dimensions: welfare and revenue. Similarly, any equilibrium with maximal total price is the "worst".

▶ **Proposition 6.** *Both welfare and revenue of equilibria are monotonically non-increasing in the total price. Therefore, an equilibrium with the minimal total price has both the highest welfare and the highest revenue, among all equilibria. Similarly, an equilibrium with the maximal total price has both the lowest welfare as well as the lowest revenue, among all equilibria.*

Proof. Consider two equilibria, one with total price v and the other with total price $w > v$. The claim that the welfare is non-increasing in the total price follows immediately from the definition. We will show that for $w > v$ it holds that $R(v) \geq R(w)$.

Lemma 5 shows that if there is an equilibrium with total price p then $(p/2, p/2)$ is also an equilibrium. As $(v/2, v/2)$ is an equilibrium, it holds that deviating to $w - v/2$ is not beneficial for a seller, and thus $R(v)/2 \geq (w - v/2)\mathcal{D}(w) \geq (w/2)\mathcal{D}(w) = R(w)/2$ and thus $R(v) \geq R(w)$ as claimed. ◀

Finally, we give a variant of a classic result by Cournot [9], which shows, somewhat counterintuitively, that a single monopolist that sells two complementary goods is better for the society than two competing sellers for each selling one of the good.

▶ **Proposition 7.** *The total price in any equilibrium is at least as high as the minimal monopolist price p^*. Thus, the welfare and revenue achieved by the monopolist price p^* are at least as high as the welfare and revenue (resp.) of the best equilibria.*

Proof. Assume that there is an equilibrium with total price $p < p^*$. As p^* is the minimal monopolist price it holds that $R(p) < R(p^*)$. Additionally, as there is an equilibrium with total price p then by Lemma 5 the pair $(p/2, p/2)$ is an equilibrium, where each seller has revenue $R(p)/2$. As $p < p^*$ a seller might deviate to $p^* - p/2 > p^*/2 > 0$, and since such deviation is not beneficial, it holds that $R(p)/2 \geq (p^* - p/2)\mathcal{D}(p^*) > (p^*/2)\mathcal{D}(p^*) = R(p^*)/2$ and thus $R(p) > R(p^*)$, a contradiction.

By Proposition 6, it follows that the welfare and revenue achieved by the minimal monopolist price p^* are no less than those in the best equilibrium. ◀

3 Existence of Non-Trivial Equilibria

In this section we show that non-trivial equilibria always exist. We first note that the structural lemmas from the previous sections seem to get us almost there: We know from Obs. 4 that the total price in equilibrium must equal one of the v_i's; We also know that if p, q is an equilibrium, then $(\frac{p+q}{2}, \frac{p+q}{2})$ is also an equilibrium. Therefore, if an equilibrium exists, then $(\frac{v_i}{2}, \frac{v_i}{2})$ must be an equilibrium for some i. However, these observations give a simple way of finding an equilibrium *if an equilibrium indeed exists*, but they do not prove existence on their own.

We give a constructive existence proof, by showing an algorithm based on an artificial dynamics that always terminates in a non-trivial equilibrium. The algorithm is essentially a sequence of best responses by the sellers, but with a twist: after every best-response step the prices are averaged. We show that this dynamics always stops at a non-trivial equilibrium and thus in particular, such equilibria exist. Moreover, when starting from prices of zero, the dynamics terminates at the best equilibrium. We formalize these claims in Proposition 10 below, from which we can clearly derive the existence of non-trivial equilibrium claimed in the next theorem as an immediate corollary.

▶ **Theorem 8.** *For any instance $(\vec{v}, \vec{d})$ there exists at least one non-trivial pure Nash equilibrium.*

Before we formally define the dynamics, we prove a simple lemma showing that the total price weakly increases as one seller best-responds to a higher price.

▶ **Lemma 9** (Monotonicity Lemma)**.** *Let $br_x \in BR(x)$ be a best reply of a seller to a price x and let $br_y \in BR(y)$ be a best reply of a seller to a price y. If $x < y \leq v_1$ then $y + br_y \geq x + br_x$.*

Proof. As $x < y \leq v_1$ by Observation 4, we know that there exists i such that $x + br_x = v_i$ and j such that $y + br_y = v_j$. As the second seller is best responding at each price level, $\mathcal{D}(v_i)(v_i - x) \geq \mathcal{D}(v_j)(v_j - x)$ and $\mathcal{D}(v_i)(v_i - y) \leq \mathcal{D}(v_j)(v_j - y)$. Together, we get that $(v_j - x)/(v_i - x) \leq \mathcal{D}(v_i)/\mathcal{D}(v_j) \leq (v_j - y)/(v_i - y)$. Now notice that the function $(a-x)/(b-x)$ is non-decreasing in x iff $a \geq b$ thus, since $y > x$, it follows that $v_j \geq v_i$. ◀

We next formally define the price-updating dynamics that we call *symmetrized best response dynamics*. It works similarly to the best response dynamics with one small difference: at each step, before a seller acts, the price of both sellers is replaced by their average price.

More formally, we start from some profile of prices (x_0, y_0). We then symmetrize the prices to $(\frac{x_0+y_0}{2}, \frac{x_0+y_0}{2})$, and then we let the first seller best reply to get prices (x_1, y_1), where $x_1 \in BR(\frac{x_0+y_0}{2})$ and $y_1 = \frac{x_0+y_0}{2}$. In one case, when the utility of the seller is 0, we need to break ties carefully: if $0 \in BR(\frac{x_0+y_0}{2})$ then we assume that $x_1 = 0$, that is, a seller with zero utility prices at 0. We then symmetrize again to $(\frac{x_1+y_1}{2}, \frac{x_1+y_1}{2})$, and then we let the second

seller best respond, symmetrize again, and continue similarly in an alternating order. The dynamic stops if the price remains unchanged in some step.

It turns out that symmetrized best response dynamics quickly converges to a non-trivial equilibrium. Moreover, we show that this dynamics is guaranteed to end up in the best equilibria. Theorem 8 follows from the following proposition.

▶ **Proposition 10.** *For any instance with n demand levels, the symmetrized best response dynamics starting with prices $(0, 0)$ reaches a non-trivial equilibrium in at most n steps, in each of them the total price increases. Moreover, this equilibrium achieves the highest social welfare and the highest revenue among all equilibria.*

Proof. We first argue that for any starting point, the sum of players' prices in the symmetrized dynamics is either monotonically increasing or monotonically decreasing. To see that, let us look at the symmetric price profiles of two consecutive steps: (x, x) and then (y, y) where $y = (x + br_x)/2$ for some $br_x \in BR(x)$ and then (z, z) where $z = (y + br_y)/2$ for some $br_y \in BR(y)$. If $x = y$, then (x, x) is an equilibrium and we are done. We first observe that if $y > x$ then $z \geq y$. Indeed, our monotonicity lemma (Lemma 9) shows exactly that: if $y > x$ then for any $br_x \in BR(x)$ and $br_y \in BR(y)$ it holds that $y + br_y \geq x + br_x$ and therefore $z \geq y$. Similarly, if $y < x$ then $z \leq y$.

To prove convergence, note that until the step where the process terminates, the total price must be either strictly increasing or strictly decreasing. Due to Observation 4, the total price at each step must be equal to v_i for some i. Since there are exactly n distinct values, the process converges after at most n steps. Note that if we reach a price level of v_n or v_1 the process must stop (no seller will have a best response that crosses these values), and a non-trivial equilibrium is reached.

Finally, we will show that a symmetrized dynamics starting at zero prices reaches an equilibrium with maximal revenue and welfare over all equilibria. Using Proposition 6, it is sufficient to show that such process reaches an equilibrium with minimum total price over all possible equilibria. This follows from the following claim:

▶ **Claim 11.** *The total price reached by a symmetrized best-response dynamics starting from a total price level x is bounded from above by the total price reached by the same dynamics starting from a total price of $y > x$,*

Proof. It is enough to show that the prices reached after a single step from x are at most those reached by a single step from y, since we can then repeat and show that this holds after all future steps. For a single step this holds due to the monotonicity lemma (Lemma 9): given some total price z, the new total price after a single step of symmetrizing the price and best responding is $f(z) = z/2 + br_{z/2}$ for some $br_{z/2} \in BR(z/2)$, and since $y > x$ it holds that $f(y) \geq f(x)$ by Lemma 9. ◀

We complete the proof by showing how the proposition follows from the last claim. Let p be the total price of the highest welfare equilibrium (lowest equilibrium price). We use the claim on total price 0 and total price $p > 0$. The symmetrized best-response dynamics starting at p stays fixed and the total price never changes, while the dynamics starting at 0 must strictly increase the total price at each step, and never go over p, and thus must end at p after at most n steps. This concludes the proof of the proposition. ◀

4 Best Response Dynamics

In the previous section we saw that non-trivial NE always exist in our price competition model, and that the best equilibrium can be easily computed. We now turn to discuss whether we can expect agents in these markets to reach such equilibria via natural adaptive heuristics. We consider the process of repeated best responses. Such a process starts from some profile of prices (p, q), then the first seller chooses a price which is a best response to q, the second seller best responds to the price chosen by the first seller, and they continue in alternating order. The process stops if no seller can improve his utility by changing his price. As we aim for non-trivial equilibria, a seller that cannot gain a positive profit chooses the best response of zero. A sequential best response process has simple and intuitive rules. The main difference between different possible dynamics of this form is in their starting prices. We will study the importance of the choice of starting prices.

Our results for best-response dynamics are negative: we show that starting from cartel prices might result in bad equilibria. We then consider starting from random prices and show that this might not help. Finally, we show that convergence time of the dynamics may be very long, even with only two demand levels.

4.1 Quality of the Dynamics' Outcomes

Probably the most natural starting prices to consider in best responses dynamics are $(0, 0)$. We start with a simple example that shows that such dynamics might result in an equilibrium with very low welfare, even when another equilibrium with high welfare exists. The gap between the quality of these equilibria is in the order of D (in the full version of the paper (see Appendix B) we show that this is the largest possible gap between equilibria).

▶ **Example 12.** Consider a market with 2 demand levels, $v_1 = 2$, $v_2 = 1$, $d_1 = 1$ and $d_2 = D$. Here, a best response dynamics starting from prices $(0, 0)$ moves to $(1, 0)$ and then ends in equilibrium prices $(1, 1)$. This NE has welfare of 2, while $(1/2, 1/2)$ is an equilibrium with welfare of $D + 1$ and revenue of D.

It follows that even with 2 demand levels, the total revenue in the highest revenue equilibrium can be factor $D/2$ larger than both the welfare and revenue of the equilibrium reached by best-response dynamics starting from prices $(0, 0)$.

One might hope that starting the dynamics from a different set of prices will guarantee convergence to a good equilibrium. Clearly, if the dynamics somehow starts from the prices of the best equilibrium it will immediately stop, but our goal is exactly to study whether the agents can adaptively reach such equilibria. One can consider two reasonable approaches for studying the starting points of the dynamics: the first approach assumes that the sellers initially agree to act as a cartel and price the bundle at the monopolist price, dividing the monopoly profit among themselves. It is well known that such a cartel is not stable, and sellers may have incentives to deviate to a different price; We would like to understand where such dynamics will stop. The second approach considers starting from a random pair of prices, and hoping that there will be a sufficient mass of starting points for which the dynamics converges to a good equilibrium. We move to study the two approaches below.

4.1.1 Dynamics Starting at a Split of the Monopolist Price

We now study best-response dynamics that start from a cartelistic solution: the total price at the starting stage is equal to the price a monopoly would have set had it owned the two

selling firms. In Example 12 we saw that splitting the monopolist price between the two sellers results in the best equilibrium. One may hope that this will generalize and such starting points ensure converging to good outcomes. In the full version we show that this is indeed the case for two demand levels. However, we next show that even with three demand levels, the welfare and revenue of the equilibrium reached by such best-response dynamics can be much lower than the revenue of the best equilibria. This holds not only when the two seller split the monopolist price evenly, but for any cartelistic split of this price. Proof can be found in the full version of this paper.

▶ **Proposition 13.** *For any large enough total demand D there is an instance with 3 demand levels and monopolist price p^* for which best response dynamics starting from any pair $(p^* - q, q)$ for $q \in [0, p^*]$, ends in an equilibrium of welfare and revenue of only 1, while there exist another equilibrium of welfare and revenue at least $\sqrt{D}/4$.*

We conclude that starting from both sellers (arbitrarily) splitting the monopolist price does not ensure that the dynamics ends in a good equilibrium, even with only 3 demand levels.

4.1.2 Dynamics Starting at Random Prices

We now consider a second approach for studying the role of starting prices in best-response dynamics. We assume that the starting prices are determined at random, and ask what are the chances that a sequence of best responses will reach a good equilibrium. Unfortunately this approach fails as well. We next show that for any $\epsilon > 0$, there is an instance with only two demand levels for which the dynamics starting from a uniform random price vector in $[0, v_1]^2$ has probability of at most ϵ of ending in an equilibrium with high welfare and revenue (although such equilibrium exists).

▶ **Proposition 14** (High probability of convergence to bad equilibria, $n = 2$)**.** *For any small enough $\epsilon > 0$ and total demand D such that $\epsilon D > 2$, there is an instance with two demand levels ($n = 2$) that has an equilibrium of welfare and revenue of at least ϵD, but best-response dynamics starting with uniform random pair of prices in $[0, v_1]^2$ ends in an equilibrium of welfare and revenue of only 1 with probability at least $1 - \epsilon$.*

Proof. Consider the input with $n = 2$ demand levels satisfying $v_1 = 1 > v_2 = \epsilon$ and $d_1 = 1 < d_2 = D$. A pair of prices (p, q) with $p + q = v_2$ results in welfare and total revenue of ϵD, and if $\epsilon D > 2$, the pair $(v_2/2, v_2/2)$ is indeed an equilibrium. On the other hand, for small enough ϵ the pair of prices $(1/2, 1/2)$ is also an equilibrium, and its welfare and revenue are only 1. Finally, observe that unless the price that the first best response in dynamics refers to is at most $v_2 = \epsilon$, the first best response results in an equilibrium with total price of 1, and welfare and revenue of 1. The probability that the process stops after a single step is therefore at least $1 - \epsilon$, and the claim follows. ◀

We show that Proposition 14 is essentially tight.

▶ **Proposition 15.** *For any instance with two demand levels for which the ratio of welfare of the best and worst equilibrium is ϵD for some $1 > \epsilon > 2/D$, it holds that the probability of the dynamics ending at the best equilibrium when starting from a uniform random pair of prices in $[0, v_1]^2$ is at least $\epsilon - 2/D$.*

Proof. Normalize the welfare of the worse equilibrium to 1 (and thus the value is 1) and the demand to 1. The best equilibrium is for demand D and value $\epsilon < 1$, since the equilibria

welfare ratio is ϵD. For any pair of prices (p, q) such that $1/D < q < \epsilon - 1/D$, the best response to q is $\epsilon - q$ as it gives revenue larger than $D \cdot (1/D) = 1$, while the maximal revenue for a seller in the other equilibrium is 1. Given price $\epsilon - q < \epsilon - 1/D$, the best response is q as it gives revenue larger than 1, while deviation will give revenue of at most 1. We conclude that with probability at least $\epsilon - 2/D$ the dynamics stops after a single best response, at the best equilibrium, as claimed. ◀

Proposition 14 only gives high probability of convergence to a low welfare equilibrium, but this will not occur with certainty. We next show that one can construct instances in which except of a measure zero set of starting prices, every dynamics will end up in an equilibrium with very low welfare, although equilibrium with high welfare exists. Moreover, we show that the welfare gap between the good and bad equilibria increases exponentially in the number of demand levels n.

▶ **Theorem 16** (Almost sure convergence to bad equilibria, large n)**.** *For any number of demand levels $n \geq 2$ and $\epsilon > 0$ that is small enough, there exists an instance that has an equilibrium with welfare $2 \cdot (2-\epsilon)^{n-1} - 1$ and revenue of $(2-\epsilon)^{n-1}$, but best response dynamics starting with pair of prices chosen uniformly at random over $[0, v_1]^2$ almost surely ends in an equilibrium of welfare and revenue of only 1.*

To prove the theorem, we build an instance where the pair of prices $(v_i/2, v_i/2)$ forms an equilibrium for any i. In this instance, the total revenue from a total price v_i is $(2-\epsilon)^{i-1}$. In particular, $(v_n/2, v_n/2)$ is an equilibrium that attains the monopolist revenue and the optimal welfare of $O((2-\epsilon)^n)$. However, best response dynamics starting by best responding to any price which is not *exactly* $v_i/2$ (for some i) terminates in an equilibrium with total price of $v_1 = 1$ and welfare of 1. Thus, the set of pairs from which the dynamics does not end at welfare of 1 is finite and has measure 0, so the dynamics almost surely converges to the worst equilibrium. The full proof is in the full version of the paper.

4.2 Time to Convergence

Up to this point we considered the quality of equilibria reached by best response dynamics. In this section, we will show that not only that best response dynamics reach equilibria of poor quality, it may also take them arbitrary long time to converge. Moreover, the long convergence time is possible even with only 2 demand levels and total demand that is close to 1.

Specifically, we will show that as the difference between the demand of adjacent values becomes smaller, the convergence time can increase. More formally, we let $W = \frac{d_n}{min_{i=2}^{n}\{d_i - d_{i-1}\}}$ be the ratio between the maximal demand and the minimal change in demand. Note that if $d_1 = 1$ and every d_i is an integer, then $d_1 = min_{i=2}^{n}\{d_i - d_{i-1}\}$ and thus $W = D$; if demands are not restricted to be integers, W might be much larger than D even in the case that $d_1 = 1$, for example if $d_1 = 1$ and $d_2 = 1 + \epsilon = D$ then $W = 1/\epsilon$ is large while $D = 1 + \epsilon \approx 1$. We show a simple setting with only two demand levels and with D close to 1 in which the dynamics takes time linear in W.

▶ **Theorem 17** (Slow convergence)**.** *For any W, best response dynamics starting from zero prices may require each seller to update his price $W - 1$ times to converge to an equilibrium. Moreover, this holds even with 2 demand levels ($n = 2$) and with $D = \frac{W}{W-1}$ which is close to 1 when W is large.*

Proof. We consider the following setting given some $\epsilon > 0$ that is small enough: $v_1 = 1$ and $d_1 = 1$, $v_2 = 1 - \epsilon$ and $d_2 = \frac{1}{1-2\epsilon}$. In this case, $W = d_2/(d_2 - d_1) = \frac{1}{2\epsilon}$. We will show that for this instance best response dynamics starting at $(0, 0)$ takes at least $W - 1 = \frac{1}{2\epsilon} - 1$ steps to converge to an equilibrium.

Let p_m, q_m denote the price offered by the two sellers after m best-response steps for each seller (p_m is the offer of the seller who plays first). We will prove by induction that $p_m = 1 - m\epsilon$ and $q_m = m\epsilon$ whenever $m + 1 < \frac{1}{2\epsilon}$.

We first handle the base case. With zero prices, the first seller can price at $v_1 = 1$ and get profit 1, or price at $v_2 = 1 - \epsilon$ and get profit $(1 - \epsilon) \cdot \frac{1}{1-2\epsilon} > 1$. Thus, $p_1 = 1 - \epsilon$. Now, the best response of the other seller is clearly $q_1 = \epsilon$ as pricing at total price of $1 - \epsilon$ gains her 0 profit.

We next move to the induction step. Assume that the claim is true for some m, i.e., $(p_m, q_m) = (1 - m\epsilon, m\epsilon)$, and we prove it for $m + 1$ (as long as $m + 1 < \frac{1}{2\epsilon}$). If the second seller prices at $m\epsilon$, the first seller will maximize profit by pricing either at $1 - (m + 1)\epsilon$ or at $1 - m\epsilon$ (recall that by Observation 4 after a seller is best responding, the price will be equal to either v_1 or v_2).

The gain from the first price is $(1 - (m + 1)\epsilon) \cdot \frac{1}{1-2\epsilon}$ and the gain from the latter price is $1 - m\epsilon$. Simple algebra shows that $(1 - (m + 1)\epsilon) \cdot \frac{1}{1-2\epsilon} > 1 - m\epsilon$ iff $m < \frac{1}{2\epsilon}$.

Now, assume that the first seller prices at $1 - (m + 1)\epsilon$, the second seller maximizes profit by pricing either at $(m + 1)\epsilon$ or at $1 - \epsilon - (1 - (m + 1)\epsilon) = m\epsilon$. The second seller chooses a price of $(m + 1)\epsilon$ if $(m + 1)\epsilon > \frac{1}{1-2\epsilon} m\epsilon$. Simple algebra shows that this holds iff $m + 1 < \frac{1}{2\epsilon}$. This concludes the induction step and completes the proof. ◀

We observe that with two demand levels, convergence to equilibrium is guaranteed, and the above linear bound is actually tight. Proof appears in the full version of the paper.

▶ **Proposition 18.** *For any instance with 2 demand levels ($n = 2$), best response dynamics starting from any price profile will stop in an equilibrium after each seller updates his price at most W times.*

5 The Quality of the Best Equilibrium

In this section, we study the price of stability in our game, that is, the ratio between the quality of the best equilibrium and the optimal outcome (both for revenue and welfare). The following theorem gives two upper bounds for the price of stability. One bound shows that for every total demand D, the best equilibrium and the optimal outcome are at most factor $O(\sqrt{D})$ away, for both welfare and revenue. The second bound is exponential in n, but it is independent of D. This implies, in particular, that the price of stability in markets with a small number of demand levels is small even for a very large D.

▶ **Theorem 19.** *For any instance, the optimal welfare and the monopolist revenue are at most $O(\min\{2^n, \sqrt{D}\})$ times the revenue of the best equilibrium.*

As the bound holds for the revenue of the best equilibrium, it clearly also holds for the welfare of that equilibrium. The proof of the theorem is in the full version of this paper.

The next theorem shows that the above price-of-stability bounds are tight. It describes instances where the gap between the best equilibrium and the optimal outcome is asymptotically at least 2^n and $\sqrt{D}$, for both welfare and revenue. We prove the theorem in the full version of the paper.

▶ **Theorem 20.** *For any number of demand levels n, there exists an instance for which the optimal welfare and the monopolist revenue are at least factor $\Omega(2^n)$ larger than the best equilibrium welfare and revenue, respectively.*

In addition, there exists an instance with integer demands for which the optimal welfare and the monopolist revenue are at least factor $\Omega(\sqrt{D})$ larger than the best equilibrium welfare and revenue, respectively.

Acknowledgements. Noam Nisan was supported by ISF grant 1435/14 administered by the Israeli Academy of Sciences and Israel-USA Bi-national Science Foundation (BSF) grant 2014389.

References

1 Elliot Anshelevich, Anirban Dasgupta, Jon M. Kleinberg, Éva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

2 Baruch Awerbuch, Yossi Azar, Amir Epstein, Vahab S. Mirrokni, and Alexander Skopalik. Fast convergence to nearly optimal solutions in potential games. In *Proceedings 9th ACM Conference on Electronic Commerce (EC08)*, pages 264–273, 2008.

3 Moshe Babaioff, Liad Blumrosen, and Noam Nisan. Network of complements. In *The 43rd International Colloquium on Automata, Languages and Programming (ICALP 2016).*, 2016.

4 Moshe Babaioff, Brendan Lucier, and Noam Nisan. Bertrand networks. In *ACM Conference on Electronic Commerce (ACM-EC)*, 2013.

5 Joseph Louis François Bertrand. theorie mathematique de la richesse sociale. *Journal de Savants*, 67:499–508, 1883.

6 James M. Buchanan and Yong J. Yoon. Symmetric tragedies; commons and anticommons. *Journal of Law and Economics*, 43(1):1–13, 2000.

7 Shuchi Chawla and Feng Niu. The price of anarchy in bertrand games. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, pages 305–314, 2009.

8 Shuchi Chawla and Tim Roughgarden. Bertrand competition in networks. In Burkhard Monien and Ulf-Peter Schroeder, editors, *Algorithmic Game Theory, First International Symposium, SAGT 2008, Paderborn, Germany, April 30-May 2, 2008. Proceedings*, volume 4997 of *Lecture Notes in Computer Science*, pages 70–82. Springer, 2008. doi:10.1007/978-3-540-79309-0_8.

9 Antoine Augustin Cournot. *Recherches sur les principes mathematiques de la theori des Richesses*. Paris : L. Hachette, 1838.

10 Nicholas Economides and Evangelos Katsamakas. Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Science*, 52(7):1057–1071, 2006.

11 Nicholas Economides and Steven C. Salop. Competition and integration among complements, and network market structure. *The Journal of Industrial Economics*, 40(1):105–123, 1992.

12 C. Ellet. *An essay on the laws of trade in reference to the works of internal improvement in the United States*. Reprints of economic classics. A.M. Kelley, 1839. URL: https://books.google.co.il/books?id=Fu4ZAAAAMAAJ.

13 Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 604–612, 2004.

14 Yossi Feinberg and Morton I. Kamien. Highway robbery: complementary monopoly and the hold-up problem. *International Journal of Industrial Organization*, 19(10):1603 – 1621, 2001.

15 Amos Fiat, Elias Koutsoupias, Katrina Ligett, Yishay Mansour, and Svetlana Olonetsky. Beyond myopic best response (in cournot competition). *Games and Economic Behavior, to appear.*, 2013.

16 Jason Hartline, Darrell Hoy, and Sam Taggart. Price of anarchy for auction revenue. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC '14, pages 693–710, 2014.

17 Michael A. Heller. The tragedy of the anticommons: Property in the transition from marx to markets. *Harvard Law Review*, 111:621 – 688, 1998.

18 E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

19 Uri Nadav and Georgios Piliouras. No regret learning in oligopolies: Cournot vs. bertrand. In Spyros C. Kontogiannis, Elias Koutsoupias, and Paul G. Spirakis, editors, *Algorithmic Game Theory - Third International Symposium, SAGT 2010, Athens, Greece, October 18-20, 2010. Proceedings*, volume 6386 of *Lecture Notes in Computer Science*, pages 300–311. Springer, 2010. doi:10.1007/978-3-642-16170-4_26.

20 Noam Nisan, Michael Schapira, Gregory Valiant, and Aviv Zohar. Best-response mechanisms. In *Innovations in Computer Science - ICS 2010*, pages 155–165, 2011.

21 Francesco Parisi, Norbert Schulz, and Ben Depoorter. Duality in property: Commons and anticommons. *International Review of Law and Economics*, 25(4):578 – 591, 2005.

22 T. Roughgarden and Eva Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236 – 259, 2002.

23 Tim Roughgarden. Intrinsic robustness of the price of anarchy. *J. ACM*, 62(5):32, 2015.

24 Alexander Skopalik and Berthold Vöcking. Inapproximability of pure nash equilibria. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 355–364, 2008.

25 Hugo Sonnenschein. The dual of duopoly is complementary monopoly: or, two of cournot's theories are one. *Journal of Political Economy*, 76:316 – 318, 1968.

Saving Critical Nodes with Firefighters is FPT*

Jayesh Choudhari[1], Anirban Dasgupta[2], Neeldhara Misra[3], and M. S. Ramanujan[4]

1 **IIT Gandhinagar, Gandhinagar, India**
choudhari.jayesh@iitgn.ac.in
2 **IIT Gandhinagar, Gandhinagar, India**
anirbandg@iitgn.ac.in
3 **IIT Gandhinagar, Gandhinagar, India**
neeldhara.m@iitgn.ac.in
4 **TU Wien, Vienna, Austria**
ramanujan@ac.tuwien.ac.at

Abstract

We consider the problem of firefighting to save a critical subset of nodes. The firefighting game is a turn-based game played on a graph, where the fire spreads to vertices in a breadth-first manner from a source, and firefighters can be placed on yet unburnt vertices on alternate rounds to block the fire. In this work, we consider the problem of saving a critical subset of nodes from catching fire, given a total budget on the number of firefighters.

We show that the problem is para-NP-hard when parameterized by the size of the critical set. We also show that it is fixed-parameter tractable on general graphs when parameterized by the number of firefighters. We also demonstrate improved running times on trees and establish that the problem is unlikely to admit a polynomial kernelization (even when restricted to trees). Our work is the first to exploit the connection between the firefighting problem and the notions of important separators and tight separator sequences.

Finally, we consider the spreading model of the firefighting game, a closely related problem, and show that the problem of saving a critical set parameterized by the number of firefighters is W[2]-hard, which contrasts our FPT result for the non-spreading model.

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

Keywords and phrases firefighting, cuts, FPT, kernelization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.135

1 Introduction

The problem of Firefighting [17] formalizes the question of designing inoculation strategies against a contagion that is spreading through a given network. The goal is to come up with a strategy for placing firefighters on nodes in order to intercept the spread of fire. More precisely, firefighting can be thought of as a turn-based game between two players, traditionally the fire and the firefighter, played on a graph G with a source vertex s. The game proceeds as follows.

- At time step 0, fire breaks out at the vertex s. A vertex on fire is said to be *burned*.
- At every odd time step $i \in \{1, 3, 5, \ldots\}$, when it is the turn of the firefighter, a firefighter is placed at a vertex v that is not already on fire. Such a vertex is permanently *protected*.
- At every even time step $j \in \{2, 4, 6, \ldots\}$, the fire spreads in the natural way: every vertex adjacent to a vertex on fire is burned (unless it was protected).

* A full version of the paper is available at https://arxiv.org/abs/1705.10923.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 135; pp. 135:1–135:13

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

The game stops when the fire cannot spread any more. A vertex is said to be *saved* if there is a protected vertex on every path from s to v. The natural algorithmic question associated with this game is to find a strategy that optimizes some desirable criteria, for instance, maximizing the number of saved vertices [4], minimizing the number of rounds, the number of firefighters per round [6], or the number of burned vertices [13, 4], and so on. These questions are well-studied in the literature, and while most variants are NP-hard, approximation and parameterized algorithms have been proposed for various scenarios. See the excellent survey [14] as well as references within for more details.

In this work, we consider the question of finding a strategy that saves a designated subset of vertices, which we shall refer to as the *critical set*. We refer to this problem as Saving A Critical Set (SACS) (we refer the reader to Section 2 for the formal definitions). This is a natural objective in situations where the goal is to save specific locations as opposed to saving some number of them. This version of the problem has been studied by [6, 18, 7] and is known to be NP-hard even when restricted to trees.

Our aim of designing firefighting solutions in order to save a critical set is well-motivated. In the context of studying networked systems for instance, it is often desirable to protect a specific set of critical infrastructure against any vulnerabilities that are cascading through the network (see [15] and [12] for an overview of *survivable network analysis* which aim to design networked systems that survive in the face of failures by providing critical services). Similarly, in the context of handling widely different risk factors that a contagion might have for different sections of the population (e.g. risk-factors that the epidemic of avian flu have for different subpopulations [5]), it is natural to ask for inoculation strategies to protect the identified at-risk groups.

Our Contributions and Methodology. We initiate the study of Saving A Critical Set from a parameterized perspective. We first show that the problem is para-NP-hard when parameterized by the size of the critical set, by showing that Saving A Critical Set is NP-complete even on instances where the size of the critical set is one. It is already clear from known results that Saving A Critical Set is para-NP-hard also when parameterized by treewidth. A third natural parameter is the number of firefighters deployed to save the critical set. Our main result is that Saving A Critical Set is FPT when parameterized by the number of firefighters, although it is not likely to have a polynomial kernel.

Our FPT algorithm is a recursive algorithm that uses the structure of tight separator sequences. The notion of tight separator sequences was introduced in [19] and has several applications [16, 20, 21] (some of which invoke modified definitions). A tight separator sequence is, informally speaking, a sequence of minimal separators such that the reachability set of S_i is contained in the reachability set of S_{i+1}. Note that any firefighting solution is a $s - C$ separator, where s is the source of the fire, and C is the critical subset of vertices. We also obtain faster algorithms on trees by using important separators.

As is common with such approaches, we do not directly solve SACS, but an appropriately generalized form, which encodes information about the behavior of some solution on the "border" vertices, which in this case is the union of all the separators in the tight separator sequence.

Related Work. The Firefighting problem has received much attention in recent years. It has been studied in the parameterized complexity setting [4, 7, 10, 2] but mostly by using the number of vertices burnt or saved as parameters. King et.al. [18] showed that for a tree of degree at most 3, it is NP-hard to save a critical set with budget of one firefighter per round,

but is polynomial time when the fire starts from a vertex of degree at most 2. Chopin [7] extended the hardness result of [18] to a per-round budget $b \geq 1$ and to trees with maximum degree $b + 2$. Chalermsook et.al.[6] gave an approximation to the number of firefighters per round when trying to protect a critical set.

Anshelevich et.al. [1] initiated the study of the spreading model, where the vaccination also spreads through the network. In Section 4 we study this problem in the parameterized setting.

2 Preliminaries

In this section, we introduce the notation and the terminology that we will need to describe our algorithms. Most of our notation is standard. We use $[k]$ to denote the set $\{1, 2, \ldots, k\}$, and we use $[k]_{\mathcal{O}}$ and $[k]_{\mathcal{E}}$, respectively, to denote the odd and even numbers in the set $[k]$.

Graphs, Important Separators and Tight Separator Sequences. We introduce here the most relevant definitions, and use standard notation pertaining to graph theory based on [9, 11]. All our graphs will be simple and undirected unless mentioned otherwise. For a graph $G = (V, E)$ and a vertex v, we use $N(v)$ and $N[v]$ to refer to the open and closed neighborhoods of v, respectively. The *distance* between vertices u, v of G is the length of a shortest path from u to v in G; if no such path exists, the distance is defined to be ∞. A graph G is said to be *connected* if there is a path in G from every vertex of G to every other vertex of G. If $U \subseteq V$ and $G[U]$ is connected, then U itself is said to be connected in G. For a subset $S \subseteq V$, we use the notation $G \setminus S$ to refer to the graph induced by the vertex set $V \setminus S$.

The following definitions about important separators and tight separator sequences will be relevant to our main FPT algorithm. We first define the notion of the reachability set of a subset X with respect to a subset S.

▶ **Definition 1** (Reachable Sets)**.** Let $G = (V, E)$ be an undirected graph, let $X \subseteq V$ and $S \subseteq V \setminus X$. We denote by $R_G(X, S)$ the set of vertices of G *reachable* from X in $G \setminus S$ and by $NR_G(X, S)$ the set of vertices of G not reachable from X in $G \setminus S$. We drop the subscript G if it is clear from the context.

We now turn to the notion of an X-Y separator and what it means for one separator to cover another.

▶ **Definition 2** (Covering by Separators)**.** Let $G = (V, E)$ be an undirected graph and let $X, Y \subset V$ be two disjoint vertex sets. A subset $S \subseteq V \setminus (X \cup Y)$ is called an $X - Y$ separator in G if $R_G(X, S) \cap Y = \emptyset$, or in other words, there is no path from X to Y in the graph $G \setminus S$. We denote by $\lambda_G(X, Y)$ the size of the smallest $X - Y$ separator in G. An $X - Y$ separator S_1 is said to *cover* an $X - Y$ separator S with respect to X if $R(X, S_1) \supset R(X, S)$. If the set X is clear from the context, we just say that S_1 covers S. An $X - Y$ separator is said to be inclusionwise minimal if none of its proper subsets is an $X - Y$ separator.

If $X = \{x\}$ is a singleton, then we abuse notation and refer to a $x - Y$ separator rather than a $\{x\} - Y$ separator. A separator S_1 dominates S if it covers S and is not larger than S in size:

▶ **Definition 3** (Dominating Separators [8])**.** Let $G = (V, E)$ be an undirected graph and let $X, Y \subset V$ be two disjoint vertex sets. An $X - Y$ separator S_1 is said to *dominate* an $X - Y$ separator S with respect to X if $|S_1| \leq |S|$ and S_1 covers S with respect to X. If the set X is clear from the context, we just say that S_1 dominates S.

We finally arrive at the notion of important separators, which are those that are not dominated by any other separator.

▶ **Definition 4** (Important Separators [8]). Let $G = (V, E)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets and $S \subseteq V \setminus (X \cup Y)$ be an $X - Y$ separator in G. We say that S is an *important* $X - Y$ separator if it is inclusionwise minimal and there does not exist another $X - Y$ separator S_1 such that S_1 dominates S with respect to X.

It is useful to know that the number of important separators is bounded as an FPT function of the size of the important separators.

▶ **Lemma 5** ([8]). *Let $G = (V, E)$ be an undirected graph, $X, Y \subset V$ be disjoint vertex sets of G. For every $k \geq 0$ there are at most 4^k important $X - Y$ separators of size at most k. Furthermore, there is an algorithm that runs in time $O(4^k k(m + n))$ which enumerates all such important X-Y separators, where $n = |V|$ and $m = |E|$.*

We are now ready to recall the notion of tight separator sequences introduced in [19]. However, the definition and structural lemmas regarding tight separator sequences used in this paper are closer to that from [21]. Since there are minor modifications in the definition as compared to the one in [21], we give the requisite proofs for the sake of completeness.

▶ **Definition 6.** Let X and Y be two subsets of $V(G)$ and let $k \in \mathbb{N}$. A *tight (X,Y)-reachability sequence* of order k is an ordered collection $\mathcal{H} = \{H_0, H_1, \ldots, H_q\}$ of sets in $V(G)$ satisfying the following properties:

- $X \subseteq H_i \subseteq V(G) \setminus N[Y]$ for any $0 \leq i \leq q$;
- $X = H_0 \subset H_1 \subset H_2 \subset \cdots \subset H_q$;
- for every $0 \leq i \leq q$, H_i is reachable from X in $G[H_i]$ and every vertex in $N(H_i)$ can reach Y in $G - H_i$
  (implying that $N(H_i)$ is a minimal (X,Y)-separator in G);
- $|N(H_i)| \leq k$ for every $1 \leq i \leq q$;
- $N(H_i) \cap N(H_j) = \emptyset$ for all $1 \leq i, j \leq q$ and $i \neq j$;
- For any $0 \leq i \leq q - 1$, there is no (X,Y)-separator S of size at most k where $S \subseteq H_{i+1} \setminus N[H_i]$ or $S \cap N[H_q] = \emptyset$ or $S \subseteq H_1$.

We let $S_i = N(H_i)$, for $1 \leq i \leq q$, $S_{q+1} = Y$, and $\mathcal{S} = \{S_0, S_1, \ldots, S_q, S_{q+1}\}$. We call $\mathcal{S}$ a *tight (X,Y)-separator sequence* of order k.

▶ **Lemma 7** (see for example [21]). *There is an algorithm that, given an n-vertex m-edge graph G, subsets $X, Y \in V(G)$ and an integer k, runs in time $O(kmn^2)$ and either correctly concludes that there is no (X,Y)-separator of size at most k in G or returns the sets $H_0, H_1, H_2 \setminus H_1, \ldots, H_q \setminus H_{q-1}$ corresponding to a tight (X,Y)-reachability sequence $\mathcal{H} = \{H_0, H_1, \ldots, H_q\}$ of order k.*

Proof. The algorithm begins by checking whether there is an X-Y separator of size at most k. If there is no such separator, then it simply outputs the same. Otherwise, it uses the algorithm of Lemma 5 to compute an arbitrary important X-Y separator S of size at most k such that there is no X-Y separator of size at most k that covers S.

Although the algorithm of Lemma 5 requires time $O(4^k k(m + n))$ to enumerate *all* important X-Y separators of size at most k, *one* important separator of the kind described in the previous paragraph can in fact be computed in time $O(kmn)$ by the same algorithm.

If there is no X-S separator of size at most k, we stop and return the set $R(X, S)$ as the only set in a tight (X, Y)-reachability sequence. Otherwise, we recursively compute a tight (X, S)-reachability sequence $\mathcal{P} = \{P_0, \ldots, P_r\}$ of order k and define $\mathcal{Q} = \{P_0, \ldots, P_r, R(X, S)\}$

as a tight (X, Y)-reachability sequence of order k. It is straightforward to see that all the properties required of a tight (X, Y)-reachability sequence are satisfied. Finally, since the time required in each step of the recursion is $O(kmn)$ and the number of recursions is bounded by n, the number of vertices, the claimed running time follows. ◀

Saving a Critical Set. We now turn to the definition of the firefighting problem. The game proceeds as described earlier: we are given a graph G with a vertex $s \in V(G)$. To begin with, the fire breaks out at s and vertex s is burning. At each step $t \geq 1$, first the firefighter protects one vertex not yet on fire - this vertex remains permanently protected - and the fire then spreads from burning vertices to all unprotected neighbors of these vertices. The process stops when the fire cannot spread anymore. In the definitions that follow, we formally define the notion of a firefighting strategy.

▶ **Definition 8** (Firefighting Strategy)**.** A k-step firefighting strategy is defined as a function $\mathfrak{h} : [2k]_{\mathcal{O}} \to V(G)$. Such a strategy is said to be *valid in G with respect to s* if, for all $i \in [2k]_{\mathcal{O}}$, when the fire breaks out in s and firefighters are placed according to $\mathfrak{h}$ for all time steps up to $i-2$, the vertex $\mathfrak{h}(i)$ is not burning at time step i, and the fire cannot spread anymore after timestep $2k$. If G and s are clear from the context, we simply say that $\mathfrak{h}$ is a valid strategy.

▶ **Definition 9** (Saving C)**.** For a vertex s and a subset $C \subseteq V(G) \setminus \{s\}$, a firefighting strategy $\mathfrak{h}$ is said to save C if $\mathfrak{h}$ is a valid strategy and $\{\mathfrak{h}(i) \mid i \in [2k]_{\mathcal{O}}\}$ is a $\{s\}$-C separator in G, in other words, there is no path from s to any vertex in C if firefighters are placed according to $\mathfrak{h}$.

We are now ready to define the parameterized problem that is the focus of this work.

Saving A Critical Set (SACS)	Parameter: k
Input:	An undirected n-vertex graph G, a vertex s, a subset $C \subseteq V(G) \setminus \{s\}$, and an integer k.
Question:	Is there a valid k-step strategy that saves C when a fire breaks out at s?

Parameterized Complexity. We follow standard terminology pertaining to parameterized algorithms based on the monograph [9]. Here we define a known technique to prove kernel lower bounds, called cross composition. Towards this, we first define polynomial equivalence relations.

▶ **Definition 10** (polynomial equivalence relation [3])**.** An equivalence relation $\mathcal{R}$ on Σ^*, where Σ is a finite alphabet, is called a *polynomial equivalence relation* if the following holds: (1) equivalence of any $x, y \in \Sigma^*$ can be checked in time polynomial in $|x| + |y|$, and (2) any finite set $S \subseteq \Sigma^*$ has at most $(\max_{x \in S} |x|)^{O(1)}$ equivalence classes.

▶ **Definition 11** (cross-composition [3])**.** Let $L \subseteq \Sigma^*$ and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L *cross-composes* into Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given t strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that: (i) $(x^*, k^*) \in Q \Leftrightarrow x_i \in L$ for some $1 \leq i \leq t$ and (ii) k^* is bounded by a polynomial in $(\max_{1 \leq i \leq t} |x_i| + \log t)$.

The following theorem allows us to rule out the existence of a polynomial kernel for a parameterized problem.

▶ **Theorem 12** ([3]). *If an* NP-hard *problem $L \subseteq \Sigma^*$ has a cross-composition into the parameterized problem Q and Q has a polynomial kernel then NP $\subseteq$ coNP/poly.*

3 The Parameterized Complexity of Saving a Critical Set

In this section, we describe the FPT algorithm for Saving A Critical Set and our cross-composition construction for trees. The starting point for our FPT algorithm is the fact that every solution to an instance (G, s, C, k) of SACS is in fact a s-C separator of size at most k. Although the number of such separators may be exponential in the size of the graph, it is a well-known fact that the number of *important* separators is bounded by $4^k n^{O(1)}$ [8]. For several problems, one is able to prove that there exists a solution that is in fact an important separator. In such a situation, an FPT algorithm is immediate by guessing the important separator.

In the SACS problem, unfortunately, there are instances where none of the solutions are important separators. However, this approach turns out to be feasible if we restrict our attention to trees, leading to improved running times. This is described in greater detail in Section 3.2. Further, in Section 3.3, we also show that we do not expect SACS to admit a polynomial kernel under standard complexity-theoretic assumptions. We establish this by a cross-composition from SACS itself, using the standard binary tree approach, similar to [2].

We describe our FPT algorithm for general graphs in Section 3.1. This is an elegant recursive procedure that operates over tight separator sequences, exploiting the fact that a solution can never be contained entirely in the region "between two consecutive separators". Although the natural choice of measure is the solution size, it turns out that the solution size by itself cannot be guaranteed to drop in the recursive instances that we generate. Therefore, we need to define an appropriate generalized instance, and work with a more delicate measure. We now turn to a detailed description of our approach.

We note that the SACS problem is para-NP-complete when parameterized by the size of the critical set, by showing that the problem is already NP-complete when the critical set has only one vertex.

▶ **Theorem 13** ($\star$). SACS *is* NP-complete *even when the critical set has one vertex.*

3.1 The FPT Algorithm

Towards the FPT algorithm for SACS, we first define a generalized firefighting problem as follows. In this problem, in addition to (G, s, C, k), we are also given the following:

- $P \uplus Q \subseteq [2k]_{\mathcal{O}}$, a set of *available time steps*,
- $Y \subset V(G)$, a subset of *predetermined firefighter locations*, and
- a bijection $\gamma : Q \to Y$, a *partial strategy* for Q.

The goal here is to find a valid partial k-step firefighting strategy over $(P \cup Q)$ that is consistent with γ on Q and saves C when the fire breaks out at s. We assume that no firefighters are placed during the time steps $[2k]_{\mathcal{O}} \setminus (P \cup Q)$. For completeness, we formally define the notion of a valid partial firefighting strategy over a set.

▶ **Definition 14** (Partial Firefighting Strategy). A *partial* k-step firefighting strategy on $X \subseteq [2k]_{\mathcal{O}}$ is defined as a function $\mathfrak{h} : X \to V(G)$. Such a strategy is said to be *valid in* G *with respect to* s if, for all $i \in X$, when the fire breaks out in s and firefighters are placed

according to $\mathfrak{h}$ for all time steps upto $[i-1]_{\mathcal{O}} \cap X$, the vertex $\mathfrak{h}(i)$ is not burning at time step i. If G and s are clear from the context, we simply say that $\mathfrak{h}$ is a valid strategy over X.

What it means for partial strategy to save C is also analogous to what it means for a strategy to save C. The only difference here is that we save C despite not placing any firefighters during the time steps j for $j \in [2k]_{\mathcal{O}} \setminus X$.

▶ **Definition 15** (Saving C with a Partial Strategy)**.** For a vertex s and a subset $C \subseteq V(G) \setminus \{s\}$, a partial firefighting strategy $\mathfrak{h}$ over X is said to save C if $\mathfrak{h}$ is a valid strategy and $\cup_{i \in X} \mathfrak{h}(i)$ is a $s-C$ separator in G, in other words, there is no path involving only burning vertices from s to any vertex in C if the fire starts at s and firefighters are placed according to $\mathfrak{h}$.

The intuition for considering this generalized problem is the following: when we recurse, we break the instance G into two parts, say subgraphs G' and H. An optimal strategy for G employs some firefighters in H at some time steps X, and the remaining firefighters in G' at time steps $[2k]_{\mathcal{O}} \setminus X$. When we recurse, we would therefore like to achieve two things:

- Capture the interactions between G' and H when we recursively solve H, so that a partial solution that we obtain from the recursion aligns with the larger graph, and
- Constrain the solution for the instance H to only use time steps in X, "allowing" firefighters to work in G' for the remaining time steps.

The constrained time steps in our generalized problem cater to the second objective, and the predetermined firefighter locations partially cater to the first. We now formally define the generalized problem.

Saving A Critical Set With Restrictions (SACS-R)	Parameter: k
Input:	An undirected n-vertex graph G, vertices s and g, a subset $C \subseteq V(G) \setminus \{s\}$, a subset $P \uplus Q \subseteq [2k]_{\mathcal{O}}$, $Y \subset V(G)$ (such that $\lvert Y\rvert = \lvert Q\rvert$, $2k-1 \in Q$ and $g \in Y$), a bijeciton $\gamma : Q \to Y$ such that $\gamma(2k-1) = g$, and an integer k.
Question:	Is there a valid partial k-step strategy over $P \cup Q$ that is consistent with γ on Q and that saves C when a fire breaks out at s?

We use p and q to denote $|P|$ and $|Q|$, respectively. Note that we can solve an SACS instance (G, s, C, k) by adding an isolated vertex g and solving the SACS-R instance $(G, s, C, 2k+2, g, P, Q, Y, \gamma)$, where $P = [2k]_{\mathcal{O}}$, $Q = \{2k+1\}$, $Y = \{g\}$ and $\gamma(2k+1) = g$. Therefore, it suffices to describe an algorithm that solves SACS-R. The role of the vertex g is mostly technical, and will be clear in due course.

We now describe our algorithm for solving an instance $\mathcal{I} := (G, s, C, k, g, P, Q, Y, \gamma)$ of SACS-R. Throughout this discussion, for the convenience of analysis of Yes instances, let $\mathfrak{h}$ be an arbitrary but fixed valid partial firefighting strategy in G over $P \cup Q$, consistent with γ on Q, that saves C. Our algorithm is recursive and works with pieces of the graph based on a tight $s-C$-separator sequence of separators of size at most $|P|$ in $G \setminus Y$. We describe the algorithm in three parts: the pre-processing phase, the generation of the recursive instances, and the merging of the recursively obtained solutions.

Phase 0 – Preprocessing. Observe that we have the following easy base cases:

- If $G \setminus Y$ has no $s-C$ separators of size at most p, then the algorithm returns No.
- If $p = 0$, then we have a Yes-instance if, and only if, s is separated from C in $G \setminus Y$ and $\mathfrak{h} := \gamma$ is a valid partial firefighting strategy over Q. In this case, the algorithm outputs Yes or No as appropriate.

- If $p > 0$ and s is already separated from C in $G \setminus Y$, then we return YES, since any arbitrary partial strategy over $P \cup Q$ that is consistent with γ on Q is a witness solution.

If we have a non-trivial instance, then our algorithm proceeds as follows. To begin with, we compute a tight $s - C$ separator sequence of order p in $G \setminus Y$. Recalling the notation of Definition 6 , we use $S_0, \ldots, S_{q+1}$ to denote the separators in this sequence, with S_0 being the set $\{s\}$ and $S_{q+1} = C$. We also use $W_0, W_1, \ldots, W_q, W_{q+1}$ to denote the reachability regions between consecutive separators. More precisely, if $\mathcal{H}$ is the tight $s - C$ reachability sequence associated with $\mathcal{S}$, then we have:

$$W_i := H_i \setminus N[H_{i-1}] \text{ for } 1 \leq i \leq q,$$

while W_{q+1} is defined as $G \setminus (N[H_q] \cup C)$. We will also frequently employ the following notation:

$$\mathcal{S} = \bigcup_{i=1}^{q} S_i \text{ and } \mathcal{W} = \bigcup_{i=1}^{q+1} W_i.$$

This is a slight abuse of notation since $\mathcal{S}$ is also used to denote the sequence $S_0, \ldots, S_{q+1}$, but the meaning of $\mathcal{S}$ will typically be clear from the context.

We first observe that if $q > k$, the separator S_q can be used to define a valid partial firefighting strategy. The intuition for this is the following: since every vertex in S_q is at a distance of at least k from s, we may place firefighters on vertices in S_q in any order during the available time steps. Since $|S_q| \leq p$ and S_q is a $s - C$ separator, this is a valid solution. Thus, we have shown the following:

▶ **Lemma 16.** *If G admits a tight $s - C$ separator sequence of order q in $G \setminus Y$ where $q > k$, then $\mathcal{I}$ is a* YES*-instance.*

Therefore, we return YES if $q > k$ and assume that $q \leq k$ whenever the algorithm proceeds to the next phase.

This concludes the pre-processing stage.

Phase 1 – Recursion. Our first step here is to guess a partition of the set of available time steps, P, into $2q + 1$ parts, denoted by $A_0, \ldots, A_q, A_{q+1}$ and $B_1, \ldots, B_{q+1}$. The partition of the time steps represents how a solution might distribute the timings of its firefighting strategy among the sets in $\mathcal{S}$ and $\mathcal{W}$. The set A_i denotes our guess of $\cup_{v \in S_i} \mathfrak{h}^{-1}(v)$ and B_j denotes our guess of $\cup_{v \in W_j} \mathfrak{h}^{-1}(v)$. Note that the number of such partitions is $(2q + 1)^p \leq (2k + 1)^k$. We define $g_0(k) := (2k + 1)^k$. We also use $\mathcal{T}_1(P)$ to denote the partition $A_0, \ldots, A_q$ and $\mathcal{T}_2(P)$ to denote $B_0, \ldots, B_{q+1}$.

We say that the partition $(\mathcal{T}_1(P), \mathcal{T}_2(P))$ is non-trivial if none of the B_i's are such that $B_i = P$. Our algorithm only considers non-trivial partitions – the reason this is sufficient follows from the way tight separator sequences are designed, and this will be made more explicit in due course.

Next, we would like to guess the behavior of a partial strategy over P restricted to $\mathcal{S}$. Informally, we do this by associating a signature with the strategy $\mathfrak{h}$, which is is a labeling of the vertex set with labels corresponding to the status of a vertex in the firefighting game when it is played out according to $\mathfrak{h}$. Every vertex is labeled as either a vertex that had a firefighter placed on it, a burned vertex, or a saved vertex. The labels also carry information about the earliest times at which the vertices attained these statuses. More formally, we have the following definition.

▶ **Definition 17.** Let $\mathfrak{h}$ be a valid k-step firefighting strategy (or a partial strategy over X). The signature of $\mathfrak{h}$ is defined as a labeling $\mathfrak{L}_\mathfrak{h}$ of the vertex set with labels from the set:

$$\mathcal{L} = (\{\mathfrak{f}\} \times X) \cup (\{\mathfrak{b}\} \times [2k]_\mathcal{E}) \cup \{\mathfrak{p}\},$$

where:

$$\mathfrak{L}_\mathfrak{h}(v) = \begin{cases} (\mathfrak{f}, t) & \text{if } \mathfrak{h}(t) = v, \\ (\mathfrak{b}, t) & \text{if } t \text{ is the earliest time step at which } v \text{ burns}, \\ \mathfrak{p} & \text{if } v \text{ is not reachable from } s \text{ in } G \setminus (\{\mathfrak{h}(i) \mid i \in [2k]_\mathcal{O}\}) \end{cases}$$

We use array-style notation to refer to the components of $\mathfrak{L}(v)$, for instance, if $\mathfrak{L}(v) = (\mathfrak{b}, t)$, then $\mathfrak{L}(v)[0] = \mathfrak{b}$ and $\mathfrak{L}(v)[1] = t$. The algorithm begins by guessing the restriction of $\mathfrak{L}_\mathfrak{h}$ on $\mathcal{S}$, that is, it loops over all possible labellings:

$$\mathfrak{T} : \mathcal{S} \to (\{\mathfrak{f}\} \times P) \cup (\{\mathfrak{b}\} \times [2k]_\mathcal{E}) \cup \{\mathfrak{p}\}.$$

The labeling $\mathfrak{T}$ is called legitimate if, for any $u \neq v$, whenever $\mathfrak{T}(u)[0] = \mathfrak{T}(v)[0] = \mathfrak{f}$, we have $\mathfrak{T}(u)[1] \neq \mathfrak{T}(v)[1]$. We say that a labeling $\mathfrak{T}$ over $\mathcal{S}$ is compatible with $\mathcal{T}_1(P) = (A_0, \ldots, A_q)$ if we have:

- for all $0 \leq i \leq r$, if $v \in S_i$ and $\mathfrak{h}(v)[0] = \mathfrak{f}$, then $\mathfrak{h}(v)[1] \in A_i$.
- for all $0 \leq i \leq r$, if $t \in A_i$, there exists a vertex $v \in S_i$ such that $\mathfrak{h}^{-1}(\mathfrak{f}, t) = v$.

The algorithm considers only legitimate labelings compatible with the current choice of $\mathcal{T}_1(P)$. By Lemma 16, we know that any tight $s - C$ separator sequence considered by the algorithm at this stage has at most k separators of size at most p each. Therefore, we have that the number of labelings considered by the algorithm is bounded by $g_1(k) := (p + k + 1)^{(kp)} \leq (3k)^{O(k^2)} \leq k^{O(k^2)}$.

We are now ready to split the graph into $q + 1$ recursive instances. For $1 \leq i \leq q + 1$, let us define $G_i = G[S_{i-1} \cup W_i \cup S_i \cup Y]$. Also, let $\mathfrak{T}_i := \mathfrak{T} \mid_{V(G_i) \cap \mathcal{S}}$. Notice that when using G_i's in recursion, we need to ensure that the independently obtained solutions are compatible with each other on the non-overlapping regions, and consistent on the common parts. We force consistency by carrying forward the information in the signature of $\mathfrak{h}$ using appropriate gadgets, and the compatibility among the W_i's is a result of the partitioning of the time steps.

Fix a partition of the available time steps P into $\mathcal{T}_1(P)$ and $\mathcal{T}_2(P)$, a compatible labeling $\mathfrak{T}$ and $1 \leq i \leq q + 1$. We will now define the SACS-R instance $\mathcal{I}\langle i, \mathcal{T}_1(P), \mathcal{T}_2(P), \mathfrak{T}_i\rangle$. Recall that $\mathcal{I} = (G, s, C, k, g, P, Q, Y, \gamma)$. To begin with, we have the following:

- Let $X_i = A_{i-1} \cup A_i$ and let $P_i = B_i$.
- Let $Q_i := X_i \cup Q$ and $Y_i := Y \cup X_i$. We define γ_i as follows:

$$\gamma_i(t) = \begin{cases} \gamma(t) & \text{if } t \in Q, \\ v & \text{if } t \in X_i \text{ and } \mathfrak{T}_i(v) = (\mathfrak{f}, t) \end{cases}$$

Note that γ_i is well-defined because the labeling was legitimate and compatible with $\mathcal{T}_1(P)$. We define H_i to be the graph $\chi(G_i, \mathfrak{T}_i)$, which is described below.

- To begin with, $V(H_i) = V(G_i) \cup \{s^\star, t^\star\}$
- Let $v \in V(G_i)$ be such that $\mathfrak{T}_i(v)[0] = \mathfrak{b}$. Use ℓ to denote $\mathfrak{T}_i(v)[1]/2$. Now, we do the following:
  - Add $k + 1$ internally vertex disjoint paths from $s^\star$ to v of length $\ell + 1$, in other words, these paths have $\ell - 1$ internal vertices.
  - Add $k + 1$ internally vertex disjoint paths from v to g of length $k - \ell - 1$.

Algorithm 1: Solve-SACS-R($\mathcal{I}$)

Input: An instance $(G, s, C, k, g, P, Q, Y, \gamma)$, $p := |P|$
Result: YES if $\mathcal{I}$ is a YES-instance of SACS-R, and NO otherwise.

```
1  if p = 0 and s and C are in different components of G \ Y then return YES;
2  else return NO;
3  if p > 0 and s and C are in different components of G \ Y then return YES;
4  if there is no s − C separator of size at most p then return NO;
5  Compute a tight s − C separator sequence S of order p.
6  if the number of separators in S is greater than k then return YES;
7  else
8      for a non-trivial partition T_1(P), T_2(P) of P into 2q + 1 parts do
9          for a labeling 𝔗 compatible with T_1(P) do
10             if ⋀_{i=1}^{q+1} (Solve-SACS-R(I⟨i, T_1(P), T_2(P), 𝔗_i⟩)) then return YES;
11     return NO
```

- Let $v \in V(G_i)$ be such that $\mathfrak{T}_i(v) = \mathfrak{p}$. Add an edge from v to $t^\star$.
- We also make $k+1$ copies of the vertices $t^\star$ and all vertices that are labeled either burned or saved. This ensures that no firefighters are placed on these vertices.

For $1 \leq i \leq q+1$, the instance $\mathcal{I}\langle i, \mathcal{T}_1(P), \mathcal{T}_2(P), \mathfrak{T}_i\rangle$ is now defined as $(\chi(G_i, \mathfrak{T}_i), s^\star, C = \{t^\star\}, k, g, P_i, Q_i, Y_i, \gamma_i)$.

Phase 2 – Merging. Our final output is quite straightforward to describe once we have the $\mathfrak{h}[\mathfrak{T}_i, i]$'s. Consider a fixed partition of the available time steps P into $\mathcal{T}_1(P)$ and $\mathcal{T}_2(P)$, and a labeling $\mathfrak{T}$ of $\mathcal{S}$ compatible with $\mathcal{T}_1(P)$. If all of the $(q+1)$ instances $\mathcal{I}\langle i, \mathcal{T}_1(P), \mathcal{T}_2(P), \mathfrak{T}_i\rangle$, $1 \leq i \leq q+1$ return YES, then we also return YES, and we return NO otherwise. Indeed, in the former case, let $\mathfrak{h}[i, \mathcal{T}_1(P), \mathcal{T}_2(P), \mathfrak{T}]$ denote a valid partial firefighting strategy for the instance $\mathcal{I}\langle i, \mathcal{T}_1(P), \mathcal{T}_2(P), \mathfrak{T}_i\rangle$. We will show that $\mathfrak{h}^\star$, described as follows, is a valid partial firefighting strategy that saves C.

- For the time steps in Q, we employ firefighters according to γ.
- For the time steps in $\mathcal{T}_1(P)$, we employ firefighters according to $\mathfrak{T}$. This is a well-defined strategy since $\mathfrak{T}$ is a compatible labeling.
- For all remaining time steps, i.e, those in $\mathcal{T}_2(P) = \{B_1, \ldots, B_{q+1}\}$, we follow the strategy given by $\mathfrak{h}[i, \mathcal{T}_1(P), \mathcal{T}_2(P), \mathfrak{T}]$.

It is easily checked that the strategy described above agrees with $\mathfrak{h}[i, \mathcal{T}_1(P), \mathcal{T}_2(P), \mathfrak{T}]$ for all i. Also, the strategy is well-defined, since $\mathcal{T}_1(P)$ and $\mathcal{T}_2(P)$ form a partition of the available time steps. Next, we will demonstrate that $\mathfrak{h}^\star$ is indeed a valid strategy that saves C, and also analyze the running time of the algorithm.

Due to lack of space, we refer the reader to the full version of this work for the analysis of the algorithm.

3.2 A Faster Algorithm For Trees

In this section we consider the setting when the input graph G is a tree. WLOG, we consider the vertex s to be the root of the tree. We first state an easy claim that shows that WLOG, we can consider the critical set to be the leaves. The proof of the following lemma follows from the fact that the firefighting solution has to be a $s - C$ separator.

▶ **Lemma 18.** *When the input graph G is a tree, if there exists a solution to* SACS*, there exists a solution such that all firefighter locations are on nodes that are on some path from s to C.*

Given the above claim, our algorithm to construct a firefighting solution is the following– exhaustively search all the important $s - C$ separators that are of size k. For each vertex v in a separator Y, we place firefighters on Y in the increasing order of distance from s and check whether this is a valid solution. The following lemma claims that if there exists a firefighting solution, the above algorithm will return one.

▶ **Lemma 19** (⋆). *Solving the* SACS *problem for input graphs that are trees takes time $O^*(4^k)$.*

3.3 No Polynomial Kernel, Even on Trees

Given that there is a FPT algorithm for SACS when restricted to trees, in this section we show that SACS on trees has no polynomial kernel. As mentioned before, the proof technique used here is on the similar lines of the proof showing no polynomial kernel for SAVING ALL BUT k-VERTICES by Bazgan et. al.[2].

▶ **Theorem 20** (⋆). SACS *when restricted to trees does not admit polynomial kernel, unless NP $\subseteq$ coNP/poly.*

4 The Spreading Model

The spreading model for firefighters was defined by Anshelevich et al. [1] as "Spreading Vaccination Model". In contrast to the firefighting game described in Section 1, in the spreading model, the firefighters (vaccination) also spread at even time steps as similar to that of the fire. That is, at any even time step if there is a firefighter at node v_i, then the firefighter extends (vaccination spreads) to all the neighbors of v_i which are not already on fire or are not already protected by a firefighter. Consider a node v_i which is not already protected or burning at time step $2j$. If u_i and w_i are neighbors of v_i, such that, u_i was already burning at time step $2j - 1$ and w_i was protected at time step $2j - 1$, then at time step $2j$, v_i is protected. That is, in the spreading model the firefighters dominate or win over fire. For the spreading model, the firefighting game can be defined formally as follows:

- At time step 0, fire breaks out at the vertex s. A vertex on fire is said to be *burned*.
- At every odd time step $i \in \{1, 3, 5, \ldots\}$, when it is the turn of the firefighter, a firefighter is placed at a vertex v that is not already on fire. Such a vertex is permanently *protected*.
- At every even time step $j \in \{2, 4, 6, \ldots\}$, first the firefighter extends to every adjacent vertex to a vertex protected by a firefighter (unless it was already protected or burned), then the fire spreads to every vertex adjacent to a vertex on fire (unless it was already protected or burned). Needless to say, the vertices protected at even time steps are also permanently *protected*.

In the following theorem, we show that in spite of the spreading power that the firefighters have, SACS is hard.

▶ **Theorem 21** (⋆). *In the spreading model,* SACS *is as hard as k-*Dominating Set.

5 Summary and Conclusions

In this work, we presented the first FPT algorithm, parameterized by the number of firefighters, for a variant of the Firefighter problem where we are interested in protecting a critical set. We also presented a faster algorithms on trees. In contrast, we also show that in the spreading model protecting a critical set is W[2]-hard. Our algorithms exploit the machinery of important separators and tight separator sequences. We believe that this opens up an interesting approach for studying other variants of the Firefighter problem.

References

1 Elliot Anshelevich, Deeparnab Chakrabarty, Ameya Hate, and Chaitanya Swamy. Approximation algorithms for the firefighter problem: Cuts over time and submodularity. In *International Symposium on Algorithms and Computation*, pages 974–983. Springer, 2009.

2 Cristina Bazgan, Morgan Chopin, Marek Cygan, Michael R Fellows, Fedor V Fomin, and Erik Jan van Leeuwen. Parameterized complexity of firefighting. *Journal of Computer and System Sciences*, 80(7):1285–1297, 2014.

3 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

4 Leizhen Cai, Elad Verbin, and Lin Yang. Firefighting on trees:$(1 - 1/e)$–approximation, fixed parameter tractability and a subexponential algorithm. In *International Symposium on Algorithms and Computation*, pages 258–269. Springer, 2008.

5 Center for Disease Control. People at High Risk of Developing Flu–Related Complications. `https://www.cdc.gov/flu/about/disease/high_risk.htm`, 2016.

6 Parinya Chalermsook and Julia Chuzhoy. Resource minimization for fire containment. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1334–1349. Society for Industrial and Applied Mathematics, 2010.

7 Morgan Chopin. *Optimization problems with propagation in graphs: Parameterized complexity and approximation*. PhD thesis, Université Paris Dauphine-Paris IX, 2013.

8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

9 Marek Cygan, Fedor V Fomin, ukasz Kowalik, Daniel Lokshtanov, D 'aniel Marx, Marcin Pilipczuk, Micha Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, Cham, Cham, 2015.

10 Marek Cygan, Fedor V Fomin, and Erik Jan van Leeuwen. Parameterized Complexity of Firefighting Revisited. In *Parameterized and exact computation*, pages 13–26. Springer, Heidelberg, Berlin, Heidelberg, 2012.

11 Reinhard Diestel. *Graph Theory*. Springer Graduate Text GTM 173. Reinhard Diestel, July 2012.

12 Robert J Ellison, David A Fisher, Richard C Linger, Howard F Lipson, and Thomas Longstaff. Survivable network systems: An emerging discipline. Technical report, DTIC Document, 1997.

13 S Finbow, B Hartnell, Q Li, and K Schmeisser. On minimizing the effects of fire or a virus on a network. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 33:311–322, 2000.

14 Stephen Finbow and Gary MacGillivray. The firefighter problem: a survey of results, directions and questions. *The Australasian Journal of Combinatorics*, 43:57–77, 2009.

15 Howard Frank and I Frisch. Analysis and design of survivable networks. *IEEE Transactions on Communication Technology*, 18(5):501–519, 1970.

16 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1670–1681, 2016.

17 Bert Hartnell. Firefighter! an application of domination. In *25th Manitoba Conference on Combinatorial Mathematics and Computing*, 1995.

18 Andrew King and Gary MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.

19 Daniel Lokshtanov and M. S. Ramanujan. Parameterized tractability of multiway cut with parity constraints. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 750–761, 2012.

20 Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. A linear time parameterized algorithm for directed feedback vertex set. *CoRR*, abs/1609.04347, 2016.

21 Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. A linear time parameterized algorithm for node unique label cover. *CoRR*, abs/1604.08764, 2016.

On the Transformation Capability of Feasible Mechanisms for Programmable Matter*†

Othon Michail[1], George Skretas[2], and Paul G. Spirakis[3]

1 **Department of Computer Science, University of Liverpool, Liverpool, UK**
Othon.Michail@liverpool.ac.uk
2 **Computer Engineering and Informatics Department, Patras University, Patras, Greece**
skretas@ceid.upatras.gr
3 **Department of Computer Science, University of Liverpool, Liverpool, UK; and Research Academic Computer Computer Technology Institute (CTI), Patras, Greece; and**
Computer Engineering and Informatics Department, Patras University, Patras, Greece
P.Spirakis@liverpool.ac.uk

Abstract

In this work, we study theoretical models of *programmable matter* systems. The systems under consideration consist of spherical modules, kept together by magnetic forces and able to perform two minimal mechanical operations (or movements): *rotate* around a neighbor and *slide* over a line. In terms of modeling, there are n nodes arranged in a 2-dimensional grid and forming some initial *shape*. The goal is for the initial shape A to *transform* to some target shape B by a sequence of movements. Most of the paper focuses on *transformability* questions, meaning whether it is in principle feasible to transform a given shape to another. We first consider the case in which only rotation is available to the nodes. Our main result is that deciding whether two given shapes A and B can be transformed to each other is in **P**. We then insist on rotation only and impose the restriction that the nodes must maintain global connectivity throughout the transformation. We prove that the corresponding transformability question is in **PSPACE** and study the problem of determining the minimum *seeds* that can make feasible otherwise infeasible transformations. Next we allow both rotations and slidings and prove universality: any two connected shapes A, B of the same number of nodes, can be transformed to each other without breaking connectivity. The worst-case number of movements of the generic strategy is $\Theta(n^2)$. We improve this to $O(n)$ parallel time, by a pipelining strategy, and prove optimality of both by matching lower bounds. We next turn our attention to distributed transformations. The nodes are now distributed processes able to perform communicate-compute-move rounds. We provide distributed algorithms for a general type of transformation.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation, F.1.3 Complexity Measures and Classes, F.2.2 Nonnumerical Algorithms and Problems, I.2.9 Robotics

Keywords and phrases programmable matter, transformation, reconfigurable robotics, shape formation, complexity, distributed algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.136

* The full version of this paper (including all omitted details and a number of helpful illustrations) can be found at: https://arxiv.org/abs/1703.04381.

† Supported in part by the School of EEE/CS of the University of Liverpool, NeST initiative.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 136; pp. 136:1–136:15

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

1 Introduction

Programmable matter refers to any type of matter that can *algorithmically* change its physical properties. For a concrete example, imagine a material formed by a collection of spherical nanomodules kept together by magnetic forces. Each module is capable of storing (in some internal representation) and executing a simple program that handles communication with nearby modules and that controls the module's electromagnets, in a way that allows the module to *rotate* or *slide* over neighboring modules. Such a material would be able to adjust its *shape* in a programmable way. Other examples of physical properties of interest for real applications would be connectivity, color [25, 5], and strength of the material.

There are already some first impressive outcomes towards the development of programmable materials (even though it is evident that there is much more work to be done in the direction of real systems), such as programmed DNA molecules that self-assemble into desired structures [28, 14] and large collectives of tiny identical robots that orchestrate resembling a single multi-robot organism (e.g., the Kilobot system [30]). Other systems for programmable matter include [21, 23]. Ambitious long-term applications of programmable materials include molecular computers, collectives of nanorobots injected into the human circulatory system for monitoring and treating diseases, or even self-reproducing and self-healing machines (see also [27]).

Apart from the fact that systems work is still in its infancy, there is also an apparent lack of unifying formalism and theoretical treatment. Still there are some first theoretical efforts aiming at understanding the fundamental possibilities and limitations of this prospective. The area of *algorithmic self-assembly* tries to understand how to program molecules (mainly DNA strands) to manipulate themselves, grow into machines and at the same time control their own growth [14]. The theoretical model guiding the study in algorithmic self-assembly is the Abstract Tile Assembly Model (aTAM) [33, 29] and variations. Recently, a model, called the *nubot* model, was proposed for studying the complexity of self-assembled structures with active molecular components [34]. Another very recent model, called the *Network Constructors* model, studied what stable networks can be constructed by a population of finite-automata that interact randomly like molecules in a well-mixed solution and can establish bonds with each other according to the rules of a common small protocol [26]. The development of Network Constructors was based on the *Population Protocol* model of Angluin *et al.* [2], that does not include the capability of creating bonds and focuses more on the computation of functions on inputs. A very interesting fact about population protocols is that they are formally equivalent to *chemical reaction networks* (CRNs), "which model chemistry in a *well-mixed solution* and are widely used to describe information processing occurring in natural cellular regulatory networks" [15]. Also the recently proposed *Amoebot* model, "offers a versatile framework to model self-organizing particles and facilitates rigorous algorithmic research in the area of programmable matter" [10, 12, 11, 13]. Other related work includes mobile and reconfigurable robotics [6, 24, 31, 20, 32, 8, 7, 4, 36, 1, 35], puzzles [9, 22], and passive systems [2, 3, 26, 19, 33, 29].

It seems that the right way for theory to boost the development of more refined real systems is to reveal the *transformation capabilities of mechanisms and technologies that are available now*, rather than by exploring the unlimited variety of theoretical models that are not expected to correspond to a real implementation in the near future. In this paper, we follow such an approach, by studying the transformation capabilities of models for programmable matter, which are based on minimal mechanical capabilities, easily implementable by existing technology.

1.1 Our Approach

We study a minimal programmable matter system consisting of n cycle-shaped modules, with each module (or *node*) occupying at any given time a cell of the 2-dimensional (abbreviated "2D" throughout) grid (no two nodes can occupy the same cell at the same time). Therefore, the composition of the programmable matter systems under consideration is discrete. Our main question throughout is whether an initial arrangement of the material can *transform* (either in principle, e.g., by an external authority, or by itself) to some other target arrangement. In more technical terms, we are provided with an *initial shape* A and a *target shape* B and we are asked whether A *can be transformed to* B via a sequence of *valid* transformation steps. Usually, a step consists either of a *valid movement* of a single node (in the *sequential case*) or of more than one nodes at the same time (in the *parallel case*). We consider two quite primitive types of movement. The first one, called *rotation*, allows a node to rotate 90° around one of its neighbors either clockwise or counterclockwise and the second one, called *sliding*, allows a node to slide by one position "over" two neighboring nodes. Both movements succeed only if the whole direction of movement is free of obstacles (i.e., other nodes blocking the way). More formal definitions are provided in Section 2. One part of the paper focuses on the case in which only rotation is available to the nodes and the other part studies the case in which both rotation and sliding are available. The latter case has been studied to some extent in the past in the, so called, *metamorphic systems* [17, 18, 16], which makes those studies the closest to our approach.

For rotation only, we introduce the notion of *color-consistency* and prove that if two shapes are not color-consistent then they cannot be transformed to each other. On the other hand, color-consistency does not guarantee transformability, as there is an infinite set of pairs (A, B) such that A and B are color consistent but still they cannot be transformed to each other. At this point, observe that if A can be transformed to B then the inverse is also true, as all movements considered in this paper are *reversible*. We distinguish two main types of transformations: those that are allowed to break the connectivity of the shape during the transformation and those that are not; we call the corresponding problems Rot-Transformability and RotC-Transformability, respectively. Our main result regarding Rot-Transformability is that Rot-Transformability $\in \mathbf{P}$. To prove polynomial-time decidability, we prove that two connected shapes A and B of the same order (i.e., having the same number of nodes) are transformable to each other iff both have at least one movement available. Therefore, transformability reduces to checking the availability of a movement in the initial and target shapes.

We next study RotC-Transformability, in which again the only available movement is rotation, but now connectivity of the material has to be preserved throughout the transformation. The property of preserving the connectivity is expected to be a crucial property for programmable matter systems, as it allows the material to maintain coherence and strength, to eliminate the need for wireless communication, and, finally, enables the development of more effective power supply schemes, in which the modules can share resources or in which the modules have no batteries but are instead constantly supplied with energy by a centralized source (or by a supernode that is part of the material itself). Such benefits can lead to simplified designs and potentially to reduced size of individual modules. We first prove that RotC-Transformability $\in$ **PSPACE**. The rest of our results here are strongly based on the notion of a *seed*. This stems from the observation that a large set of infeasible transformations become feasible by introducing to the initial shape an additional, and usually quite small, seed; i.e., a small shape that is being attached to some point of the initial shape. We investigate seeds that could serve as components capable of traveling the

perimeter of an arbitrary connected shape A. Such seed-shapes are very convenient as they are capable of "simulating" the *universal transformation* techniques that are possible if we have both rotation and sliding movements available (discussed in the sequel). To this end, we prove that all seeds of size ≤ 4 cannot serve for this purpose, by proving that they cannot even walk the perimeter of a simple line shape. On the other hand, we manage to show that a *6-seed* succeeds, and this provides a first indication, that there might be a large family of shapes that can be transformed to each other with rotation only and without breaking connectivity.

Next, we consider the case in which both rotation and sliding are available and insist on connectivity preservation. We first provide a proof that this combination of simple movements is *universal* w.r.t. transformations, as any pair of connected shapes A and B of the same order can be transformed to each other without ever breaking the connectivity throughout the transformation (a first proof of this fact had already appeared in [16]). This generic transformation requires $\Theta(n^2)$ sequential movements in the worst case. By a potential-function argument we show that no transformation can improve on this worst-case complexity for some specific pairs of shapes and this lower bound is independent of connectivity preservation; it only depends on the inherent *transformation-distance* between the shapes. To improve on this, either some sort of parallelism must be employed or more powerful movement mechanisms, e.g., movements of whole sub-shapes in one step. We investigate the former approach, and prove that there is a *pipelining* general transformation strategy that improves the time to $O(n)$ (parallel time). We also give a matching $\Omega(n)$ lower bound. On the way, we also show that this parallel complexity is feasible even if the nodes are labeled, meaning that individual nodes must end up in specific positions of the target-shape.

Finally, we assume that the nodes are distributed processes able to perform communicate-compute-move rounds (where, again, both rotation and sliding movements are available) and provide distributed algorithms for a general type of transformation.

Section 2 brings together all definitions and basic facts that are used throughout the paper. In Section 3, we study programmable matter systems equipped only with rotation movement. In Section 4, we insist on rotation only, but additionally require that the material maintains connectivity throughout the transformation. In Section 5, we investigate the combined effect of rotation and sliding movements. Finally, in Section 6 we conclude and give further research directions that are opened by our work.

2 Preliminaries

The programmable matter systems considered in this paper operate on a 2D square grid, with each position (or *cell*) being uniquely referred to by its $y \geq 0$ and $x \geq 0$ coordinates. Such a system consists of a set V of n *modules*, called *nodes* throughout. Each node may be viewed as a spherical module fitting inside a cell of the grid. At any given time, each node $u \in V$ occupies a cell $o(u) = (o_y(u), o_x(u)) = (i, j)$ (where i corresponds to a row and j to a column of the grid) and no two nodes may occupy the same cell. At any given time t, the positioning of nodes on the grid defines an undirected *neighboring relation* $E(t) \subset V \times V$, where $\{u, v\} \in E$ iff $o_y(u) = o_y(v)$ and $|o_x(u) - o_x(v)| = 1$ or $o_x(u) = o_x(v)$ and $|o_y(u) - o_y(v)| = 1$, that is, if u and v are either *horizontal* or *vertical* neighbors on the grid, respectively. A more informative way to define the system at a given time t, and thus often more convenient, is as a mapping $P_t \colon \mathbb{N}_{\geq 0} \times \mathbb{N}_{\geq 0} \to \{0, 1\}$ where $P_t(i, j) = 1$ iff cell (i, j) is occupied by a node.

At any given time t, $P_t^{-1}(1)$ defines a *shape*. Such a shape is called *connected* if $E(t)$ defines a connected graph. A connected shape is called *convex* if for any two occupied cells,

the line that connects their centers does not pass through an empty cell. We call a shape *discrete-convex* if for any two occupied cells, belonging either to the same row or the same column, the line that connects their centers does not pass through an empty cell; i.e., in the latter we exclude diagonal lines. We call a shape *compact* if it has no holes.

In general, shapes can *transform* to other shapes via a sequence of one or more *movements* of individual nodes. Time consists of discrete *steps* (or *rounds*) and in every step, zero or more movements may occur. In the *sequential* case, at most one movement may occur per step, and in the *parallel* case any number of "valid" movements may occur in parallel. [1] We consider two types of movements: (i) *rotation* and (ii) *sliding*. In both movements, a single node moves relative to one or more neighboring nodes as we just explain.

A single *rotation* movement of a node u is a 90° *rotation* of u around one of its neighbors. Let (i, j) be the current position of u and let its neighbor be v occupying the cell $(i-1, j)$ (i.e., lying below u). Then u can *rotate* 90° clockwise (counterclockwise) around v iff the cells $(i, j+1)$ and $(i-1, j+1)$ ($(i, j-1)$ and $(i-1, j-1)$, respectively) are both empty. By rotating the whole system 90°, 180°, and 270°, all possible rotation movements are defined analogously. A single *sliding* movement of a node u is a one-step horizontal or vertical movement "over" a horizontal or vertical line of (neighboring) nodes of length 2. In particular, if (i, j) is the current position of u, then u can *slide* rightwards to position $(i, j+1)$ iff $(i, j+1)$ is not occupied and there exist nodes at positions $(i-1, j)$ and $(i-1, j+1)$ or at positions $(i+1, j)$ and $(i+1, j+1)$, or both. Precisely the same definition holds for up, left, and down sliding movements by rotating the whole system 90°, 180°, and 270° counterclockwise, respectively.

Let A and B be two shapes. We say that A *transforms to* B *via a movement* m (which can be either a rotation or a sliding), denoted $A \xrightarrow{m} B$, if there is a node u in A such that if u applies m, then the shape resulting after the movement is B (possibly after rotations and translations of the resulting shape, depending on the application). We say that A *transforms in one step to* B (or that B *is reachable in one step from* A), denoted $A \to B$, if $A \xrightarrow{m} B$ for some movement m. We say that A *transforms* to B (or that B is *reachable* from A) and write $A \rightsquigarrow B$, if there is a sequence of shapes $A = C_0, C_1, \ldots, C_t = B$, such that $C_i \to C_{i+1}$ for all i, $0 \leq i < t$. We should mention that we do not always allow m to be any of the two possible movements. In particular, in Sections 3 and 4 we only allow m to be a rotation, as we there restrict attention to systems in which only rotation is available. We shall clearly explain what movements are permitted in each part of the paper. Observe now that both rotation and sliding are *reversible* movements, a fact that we extensively use in our results. Based on this, it can be proved that the relation '$\rightsquigarrow$' is a partial equivalence relation. When the only available movement is rotation, there are shapes in which no rotation can be performed. If we introduce a *null* rotation, then every shape may transform to itself by applying the *null* rotation, and '$\rightsquigarrow$' becomes an equivalence relation.

The following are the main transformation problems that are considered in this work:

Rot-Transformability. Given an initial shape A and a target shape B (usually both connected), decide whether A can be transformed to B (usually, under translations and rotations of the shapes) using only a sequence of rotation movements.

RotC-Transformability. Special case of Rot-Transformability, where A and B are connected shapes and connectivity must be preserved throughout the transformation.

RS-Transformability. Variant of Rot-Transformability in which both rotation and sliding movements are available.

[1] By "valid", we mean here subject to the constraint that their whole movement paths correspond to pairwise disjoint sub-areas of the grid.

Minimum-Seed-Determination. Given an initial shape A and a target shape B determine a minimum-size seed and an initial positioning of that seed relative to A that makes the transformation from A to B feasible.

3 Rotation

In this section, the only permitted movement is 90° *rotation* around a neighbor. Our main result in this section is that Rot-Transformability $\in \mathbf{P}$.

Consider a black and red checkered coloring of the 2D grid. Any shape S may be viewed as a colored shape consisting of $b(S)$ blacks and $r(S)$ reds. Call two shapes A and B *color-consistent* if $b(A) = b(B)$ and $r(A) = r(B)$ and call them *color-inconsistent* otherwise. Call a transformation from a shape A to a shape C *color-preserving* if A and C are color consistent.

▶ **Observation 1.** *The rotation movement is color-preserving. Formally, $A \rightsquigarrow C$ (restricted to rotation only) implies that A and C are color-consistent. In particular, every node beginning from a black (red) position of the grid, will always be on black (red, respectively) positions throughout a transformation.*

Based on this property of the rotation movement, we may call each node *black* or *red* throughout a transformation, based only on its initial coloring. Observation 1 gives a partial way to determine that two shapes A and B cannot be transformed to each other by rotations.

▶ **Proposition 2.** *If two shapes A and B are color-inconsistent, then it is impossible to transform one to the other by rotations only.*

▶ **Proposition 3.** *There is a generic connected shape, called* line-with-leaves*, that has a color-consistent version for any connected shape A.*

Proof. Let red be the majority color of A and k be the number of black nodes of A. Consider a bi-color line starting with a black node and ending to a black node, such that all k blacks are exhausted. To do this, $k-1$ reds are needed in order to alternate blacks and reds on the line. Since A is connected, it can have at most $3k+1$ reds. By adding red leaf-nodes around the blacks of the line, we can achieve the whole range of possible number of reds, from k to $3k+1$. ◀

Based on this, we now show that the inverse of Proposition 2 is not true, that is, it does not hold that any two color-consistent shapes can be transformed to each other by rotations.

▶ **Proposition 4.** *There is an infinite set of pairs (A, B) of connected shapes, such that A and B are color-consistent but cannot be transformed to each other by rotations only.*

Proof. For shape A, take a rhombus in which no node is able to rotate. By Proposition 3, any such A has a color-consistent shape B from the family of line-with-leaves shapes, such that $B \neq A$. We conclude that A and B are distinct color-consistent shapes which cannot be transformed to each other, and there is an infinite number of such pairs, as the number of black nodes of A can be made arbitrarily large. ◀

Propositions 2 and 4 give a partial characterization of pairs of shapes that cannot be transformed to each other. Observe that the impossibilities proved so far, hold for all possible transformations based on rotation only, including those that are allowed to break connectivity.

The next theorem states that the inclusion between RotC-Transformability and Rot-Transformability is strict, that is, there are strictly more feasible transformations if we allow connectivity to break. We prove this by showing that there is a feasible

transformation, namely folding a spanning line in half, in Rot-Transformability\RotC-Transformability.

▶ **Theorem 5.** RotC-Transformability ⊂ Rot-Transformability.

Aiming at a general transformation, we ask whether there is some minimal addition to a shape that would allow it to transform. The solution turns out to be as small as a *2-line seed* (a bi-color pair, usually referred to as "2-line" or "2-seed") lying initially somewhere "outside" the boundaries of the shape. Based on the above assumptions, we prove that any pair of color-consistent connected shapes A and B can be transformed to each other. The idea is to exploit the fact that the 2-line can move freely in any direction and to use it in order to extract from A another 2-line. In this way, a 4-line seed is formed, which can also move freely in all directions. Then we use the 4-line as a transportation medium for carrying the nodes of A, one at a time. We exploit these mobility mechanisms to transform A into a uniquely defined shape from the line-with-leaves family of Proposition 3. But if any connected shape A with an extra 2-line can be transformed to its color-consistent line-with-leaves version with an extra 2-line, then this also holds inversely due to reversibility, and it follows that any A can be transformed to any B by transforming A to its line-with-leaves version L_A and then inverting the transformation from B to $L_B = L_A$.

▶ **Theorem 6.** *If connectivity can break and there is a 2-line seed provided "outside" the initial shape, then any pair of color-consistent connected shapes A and B can be transformed to each other by rotations only.*

Proof. Without loss of generality (due to symmetry and the 2-line's unrestricted mobility), it suffices to assume that the seed is provided somewhere below the lowest row l occupied by the shape A. We show how A can be transformed to L_A with the help of the seed. We define L_A as follows: Let k be the cardinality of the minority color, let it be the black color. As there are at least k reds, we can create a horizontal line of length $2k$, i.e., $u_1, u_2, \ldots, u_{2k}$, starting with a black (i.e., u_1 is black), and alternating blacks and reds. In this way, the blacks are exhausted. The remaining $\leq (3k+1) - k = 2k+1$ reds are then added as leaves of the black nodes, starting from the position to the left of u_1 and continuing counterclockwise, i.e., below u_1, below u_3, ..., below u_{2k-1}, above u_{2k-1}, above u_{2k-3}, and so on. This gives the same shape from the line-with-leaves family, for all color-consistent shapes (observe that the leaf to the right of the line is always placed). L_A shall be constructed on rows $l-5$ to $l-3$ (not necessarily inclusive), with u_1 on row $l-4$ and a column j preferably between those that contain A.

First, extract a 2-line from A, from row l, so that the 2-line seed becomes a 4-line seed. To see that this is possible for every shape A of order at least 2, distinguish the following two cases: (i) If the lowest row has a horizontal 2-line, then the 2-line can leave the shape without any help and approach the 2-seed. (ii) If not, then take any node u of row l. As A is connected and has at least two nodes, u must have a neighbor v above it. The only possibility that the 2-line u,v is not free to leave A is when v has both a left and a right neighbor, but this can be resolved with the help of the 2-line.

To transform A to L_A, given the 4-line seed, do the following:

- While blacks is still present in A:
  - If on the current lowest row occupied by A, there is a 2-line that can be extracted alone and moved towards L_A, then perform the shortest such movement that attaches the 2-line to the right endpoint of L_A's line $u_1, u_2, \ldots$.
  - If not, then do the following. Maintain a *repository* of nodes at the empty space below row $l-7$, initially empty. If, either in the lowest row of A or in the repository, there

is a node of opposite color than the current color of the right endpoint of L_A's line, use the 4-line to transfer such a node and make it the new right endpoint of L_A's line. Otherwise, use the 4-line to transfer a node of the lowest row of A to the repository.

- Once black has been exhausted from A and the repository (i.e., when u_{2k-3} has been placed; u_{2k-1} and u_{2k} will only be placed in the end as they are part of the 4-line), transfer a red to position u_{2k-2}. If there are no more nodes left, run the termination phase, otherwise transfer the remaining nodes (all red) with the 4-line, one after the other, and attach them as leaves around the blacks of L_A's line, beginning from the position to the left of u_1 counterclockwise, as described above (skipping position u_{2k}).
- Termination phase: the line-with-leaves is ready, apart from positions u_{2k-1}, u_{2k} which require a 2-line from the 4-line. If the position above u_{2k-1} is empty, then extract a 2-line from the 4-line and transfer it to the positions u_{2k-1}, u_{2k}. This completes the transformation. If the position above u_{2k-1} is occupied by a node u_{2k+1}, then place the whole 4-line vertically with its lowest endpoint on u_{2k}. Then rotate the top endpoint counterclockwise, to move above u_{2k+1}, then rotate u_{2k+1} clockwise around it to move to its left, then rotate the node above u_{2k} counterclockwise to move to u_{2k-1}, and finally restore u_{2k+1} to its original position. This completes the construction (the 2-line that always remains can be transferred in the end to a predefined position).

◀

The natural next question is to what extent the 2-line seed assumption can be dropped. Clearly, by Proposition 4, this cannot be always possible. The following lemma gives a sufficient and necessary condition for dropping the 2-line seed assumption.

▶ **Lemma 7.** *A 2-seed can be extracted from a shape iff a single rotation move is available on the shape.*

▶ **Theorem 8.** Rot-Transformability $\in \mathbf{P}$.

Proof. If the two connected input shapes of the same order are not already equal, then, by Lemma 7 and Theorem 6, it suffices to check if both shapes have an available movement. If yes, *accept*, otherwise, *reject*. These checks can be easily performed in polynomial time. ◀

4 Rotation and Connectivity Preservation

In this section, we restrict our attention to transformations that transform a connected shape A to one of its color-consistent connected shapes B, without ever breaking the connectivity of the shape on the way. As already mentioned in the introduction, connectivity preservation is a very desirable property for programmable matter, as, among other positive implications, it guarantees that communication between all nodes is maintained, it minimizes transformation failures, requires less sophisticated actuation mechanisms, and increases the external forces required to break the system apart.

We begin by proving that RotC-Transformability can be decided in deterministic polynomial space.

▶ **Theorem 9.** RotC-Transformability $\in$ **PSPACE**.

As already shown in Theorem 5, the connectivity-preservation constraint increases the class of infeasible transformations. A convenient turnaround in such cases, is to introduce a suitable seed that can assist the transformation. For example, we can circumvent the impossibility of folding a line $u_1, u_2, \ldots, u_n$ in half, by adding a 3-line seed v_1, v_2, v_3, horizontally aligned

over nodes u_3, u_4, u_5 of the line. Interestingly, adding the seed over nodes u_4, u_5, u_6 does not work. Therefore, the problem that we face in such cases, is to find a minimum seed (could be any connected small shape, not necessarily a line) and a placement of that seed, that enables the otherwise infeasible transformation (*Minimum-Seed-Determination* problem). In the rest of this section, we try to identify a minimum seed that can walk the perimeter of any shape, hoping that it will be able to move nodes gradually to a predetermined position, in order to transform the initial shape into a line-with-leaves (as in Theorem 6, but without ever breaking connectivity this time). [2]

▶ **Theorem 10.** *If connectivity must be preserved:* (i) *Any* (≤ 4)*-seed cannot traverse the perimeter of a line,* (ii) *a 6-seed can traverse the perimeter of any discrete-convex shape.*

5 Rotation and Sliding

In this section, we study the combined effect of rotation and sliding movements. We begin by proving that rotation and sliding together are *transformation-universal*, meaning that they can transform any given shape to any other shape of the same size without ever breaking the connectivity during the transformation.

▶ **Theorem 11.** *Let A and B be any connected shapes, such that $|A| = |B| = n$. Then A and B can be transformed to each other by rotations and slidings, without breaking the connectivity during the transformation.*

Proof. It suffices to show that any connected shape A can be transformed to a spanning line L using only rotations and slidings and without breaking connectivity during the transformation. If we show this, then A can be transformed to L and B can be transformed to L (as A and B have the same order, therefore corresponding to the same spanning line L), and by reversibility of these movements, A and B can be transformed to each other via L.

Pick the rightmost column of the grid containing at least one node of A, and consider the lowest node of A in that column. Call that node u. Observe that all cells to the right of u are empty. Let the cell of u be (i, j). The final constructed line will start at (i, j) and end at $(i, j + n - 1)$.

The transformation is partitioned into $n - 1$ phases. In each phase k, we pick a node from the original shape and move it to position $(i, j + k)$, that is, to the right of the right endpoint of the line formed so far. In phase 1, position $(i, j + 1)$ is a cell of the perimeter of A. So, even if it happens that u is a node of degree 1, it can be proved that there must be another such node $v \in A$ that can walk the whole perimeter of $A' = A - \{v\}$. As $u \neq v$, $(i, j + 1)$ is also part of the perimeter of A', therefore, v can move to $(i, j + 1)$ by rotations and slidings. But A' is connected, $A' \cup \{(i, j + 1)\}$ is also connected, and also all intermediate shapes were connected, because v moved on the perimeter and, therefore, it never disconnected from the rest of the shape during its movement.

In general, the transformation preserves the following invariant. At the beginning of phase k, $1 \leq k \leq n - 1$, there is a connected shape $S(k)$ (where $S(1) = A$) to the left of of column j (j inclusive) and a line of length $k - 1$ starting from position $(i, j + 1)$ and growing to the right. Restricting attention to $S(k)$, there is always a $v \neq u$ that could (hypothetically) move to position $(i, j + 1)$ if it were not occupied. This implies that before the final movement that

[2] Another way to view this, is as an attempt to simulate the universal transformations based on combined rotation and sliding (presented in Section 5), in which single nodes are able to walk the perimeter of the shape.

would place v on $(i, j+1)$, v must have been in $(i+1, j)$ or $(i+1, j+1)$, if we assume that v always walks in the clockwise direction. Observe now that from each of these positions v can perform zero or more right slidings above the line in order to reach the position above the right endpoint $(i, j+k-1)$ of the line. When this occurs, a final clockwise rotation makes v the new right endpoint of the line. The only exception is when v is on $(i+1, j+1)$ and there is no line to the right of (i, j) (this implies the existence of a node on $(i+1, j)$, otherwise connectivity of $S(k)$ would have been violated). In this case, v just performs a single downward sliding to become the right endpoint of the line. ◀

▶ **Theorem 12.** *The transformation of Theorem 11 requires $\Theta(n^2)$ movements in the worst case.*

Theorem 12 shows that the above generic strategy is slow in some cases, as is the case of transforming a staircase shape into a spanning line. A *staircase* is defined as a shape of the form $(i, j), (i-1, j), (i-1, j+1), (i-2, j+1), (i-2, j+2), (i-3, j+2), \ldots$. We shall now show that there are pairs of shapes for which any strategy and not only this particular one, may require a quadratic number of steps to transform one shape to the other.

▶ **Definition 13.** Define the *potential of a shape A* as its minimum "distance" from the line L, where $|A| = |L|$. The *distance* is defined as follows: Consider any placement of L relative to A and any pairing of the nodes of A to the nodes of the line. Then sum up the Manhattan distances [3] between the nodes of each pair. The minimum sum between all possible relative placements and all possible pairings is the distance between A and L and also A's potential.

Observe that the potential of the line is 0 as it can be totally aligned on itself and the sum of the distances is 0.

▶ **Lemma 14.** *The potential of a staircase is $\Theta(n^2)$.*

Proof. We prove it for horizontal placement of the line, as the vertical case is symmetric. Any such placement leaves either above or below it at least half of the nodes of the staircase (maybe minus 1). W.l.o.g. let it be above it. Every two nodes, the height increases by 1, therefore there are 2 nodes at distance 1, 2 at distance 2,..., 2 at distance n/4. Any matching between these nodes and the nodes of the line gives for every pair a distance at least as large as the vertical distance between the staircase's node and the line, thus, the total distance is at least $2 \cdot 1 + 2 \cdot 2 + ... + 2 \cdot (n/4) = 2 \cdot (1 + 2 + ... + n/4) = (n/4) \cdot (n/4 + 1) = \Theta(n^2)$. We conclude that the potential of the staircase is $\Theta(n^2)$. ◀

▶ **Theorem 15.** *Any transformation strategy based on rotations and slidings which performs a single movement per step requires $\Theta(n^2)$ steps to transform a staircase into a line.*

Proof. To show that $\Omega(n^2)$ movements are needed to transform the staircase into a line, it suffices to observe that the difference in their potentials is that much and that one rotation or one sliding can decrease the potential by at most 1. ◀

▶ Remark. The above lower bound is independent of connectivity preservation. It is just a matter of the total distance based on single distance-one movements.

Finally, it is interesting to observe that such lower bounds can be computed in polynomial time, because there is a polynomial-time algorithm for computing the distance between two shapes.

---

[3] The Manhattan distance between two points (i, j) and (i', j') is given by $|i - i'| + |j - j'|$.

▶ **Proposition 16.** *Let A and B be connected shapes. Then their distance $d(A, B)$ can be computed in polynomial time.*

To give a faster transformation either pipelining must be used (allowing for more than one movement in parallel) or more complex mechanisms that move sub-shapes consisting of many nodes, in a single step. We follow the former approach, by allowing an unbounded number of rotation and/or sliding movements to occur simultaneously in a single step (though, in pairwise disjoint areas).

▶ **Proposition 17.** *There is a pipelining strategy that transforms a staircase into a line in $O(n)$ parallel time.*

Proof. Number the nodes of the staircase 1 through n starting from the top and following the staircase's connectivity until the bottom-right node is reached. These gives an odd-numbered upper diagonal and an even-numbered lower diagonal. Node 1 moves as in Theorem 11. Any even node w starts moving as long as its upper odd neighbor has reached the same level as w (e.g., node 2 first moves after node 1 has arrived to the right of node 3). Any odd node $z > 1$ starts moving as long as its even left neighbor has moved one level down (e.g., node 3 first moves after node 2 has arrived to the right of 5). After a node starts moving, it moves in every step as in Theorem 11 (but now many nodes can move in parallel, implementing a pipelining strategy). It can be immediately observed that any node i starts after at most 3 movements of node $i - 1$ (actually, only 2 movements for even i), so after, roughly, at most $3n$ steps, node $n - 2$ starts. Moreover, a node that starts, arrives at the right endpoint of the line after at most n steps, which means that after at most $4n = O(n)$ steps, all nodes have taken their final position in the line. ◀

Proposition 17 gives a hint that pipelining could be a general strategy to speed-up transformations. We next show how to generalize this technique to any possible pair of shapes.

▶ **Theorem 18.** *Let A and B be any connected shapes, such that $|A| = |B| = n$. Then there is a pipelining strategy that can transform A to B (and inversely) by rotations and slidings, without breaking the connectivity during the transformation, in $O(n)$ parallel time.*

Proof. The transformation is a pipelined version of the sequential transformation of Theorem 11. Now, instead of picking an arbitrary next candidate node of $S(k)$ to walk the perimeter of $S(k)$ clockwise, we always pick the rightmost clockwise node $v_k \in S(k)$, that is, the node that has to walk the shortest clockwise distance to arrive at the line being formed. This implies that the subsequent candidate node v_{k+1} to walk is always "behind" v_k in the clockwise direction and is either already free to move or is enabled after v_k's departure. Observe that after at most 3 clockwise movements, v_k can no longer be blocking v_{k+1} on the (possibly updated) perimeter. Moreover, the clockwise move of v_{k+1} only introduces a gap in its original position, therefore it only affects the structure of the perimeter "behind" it. The strategy is to start the walk of node v_{k+1} as soon as v_k is no longer blocking its way. As in Proposition 17, once a node starts, it moves in every step, and again any node arrives at the end of the forming line after at most n movements. It follows that if the pipelined movement of nodes cannot be blocked in any way, after $4n = O(n)$ steps all nodes must have arrived at their final positions. Observe now that the only case in which pipelining could be blocked is when a node is sliding through a (necessarily dead-end) "tunnel" of height 1. To avoid this, the nodes shortcut the tunnel, by visiting only its first position (i, j) and then simply skipping the whole walk inside it (that walk would just return them to position (i, j) after a number of steps). ◀

We next show that even if A and B are labeled shapes, that is, their nodes are assigned the indices $1, \ldots, n$ (uniquely, i.e., without repetitions), we can still transform the labeled A to the labeled B with only a linear increase in parallel time. We only consider transformations in which the nodes never change indices in any way (e.g., cannot transfer them, or swap them), so that each particular node of A must eventually occupy (physically) a particular position of B (the one corresponding to its index).

▶ **Corollary 19.** *The labeled version of the transformation of Theorem 18 can be performed in $O(n)$ parallel time.*

An immediate observation is that a linear-time transformation does not seem satisfactory for all pairs of shapes. To this end, take a square S and rotate its top-left corner u, one position clockwise, to obtain an almost-square S'. Even though, a single counter-clockwise rotation of u suffices to transform S' to S, the transformation of Theorem 18 may go all the way around and first transform S' to a line and then transform the line to S. In this particular example, the distance between S and S', according to Definition 13, is 2, while the generic transformation requires $\Theta(n)$ parallel time. So, it is plausible to ask if any transformation between two shapes A and B can be performed in time that grows as a function of their distance $d(A, B)$. We show that this cannot always be the case, by presenting two shapes A and B with $d(A, B) = 2$, such that A and B require $\Omega(n)$ parallel time to be transformed to each other.

▶ **Proposition 20.** *There are two shapes A and B with $d(A, B) = 2$, such that A and B require $\Omega(n)$ parallel time to be transformed to each other.*

In the full version, we also study the RS-Transformability problem in distributed systems and give an algorithm that transforms a large family of shapes into a spanning line:

▶ **Theorem 21.** *We provide an algorithm, called* Compact Line*, that can transform any compact shape into a spanning line.*

6 Conclusions and Further Research

There are many open problems related to the findings of the present work. First, a compromise could be to allow some restricted degree of connectivity breaking. There are other meaningful "good" properties that we would like to maintain throughout a transformation, like the *strength* of the shape.

Transformation seems in general harder if we restrict the maximum area or dimensions during its course. Also, restricting the boundaries gives models equivalent to several interesting puzzles, like the famous 15-puzzle. Techniques developed in the context of puzzles could prove valuable for analyzing and characterizing discrete programmable matter systems.

We intentionally restricted attention to very minimal actuation mechanisms. More sophisticated mechanical operations would enable a larger set of transformations and possibly also reduce the time complexity. Such an example is the ability of a node to become inserted between two neighboring nodes.

There are also some promising specific technical questions: What is the exact complexity of RotC-Transformability? What is the complexity of computing the optimum transformation? Can it be satisfactorily approximated? Finally, regarding the distributed transformations, there are various interesting variations of the model considered here, that would make sense. One of them is to assume nodes that are oblivious w.r.t. their orientation.

References

1 Greg Aloupis, Nadia Benbernou, Mirela Damian, Erik D. Demaine, Robin Flatland, John Iacono, and Stefanie Wuhrer. Efficient reconfiguration of lattice-based modular robots. *Computational geometry*, 46(8):917–928, 2013.

2 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, March 2006.

3 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, November 2007.

4 Zack Butler, Keith Kotay, Daniela Rus, and Kohji Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *The International Journal of Robotics Research*, 23(9):919–937, 2004.

5 Xuli Chen, Li Li, Xuemei Sun, Yanping Liu, Bin Luo, Changchun Wang, Yuping Bao, Hong Xu, and Huisheng Peng. Magnetochromatic polydiacetylene by incorporation of Fe3O4 nanoparticles. *Angewandte Chemie International Edition*, 50(24):5486–5489, 2011.

6 Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Solving the robots gathering problem. In *International Colloquium on Automata, Languages, and Programming*, pages 1181–1196. Springer, 2003.

7 Alejandro Cornejo, Fabian Kuhn, Ruy Ley-Wild, and Nancy Lynch. Keeping mobile robot swarms connected. In *Proceedings of the 23rd international conference on Distributed computing*, DISC'09, pages 496–511, Berlin, Heidelberg, 2009. Springer-Verlag.

8 Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, 28(2):131–145, April 2015.

9 Erik D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *International Symposium on Mathematical Foundations of Computer Science*, pages 18–33. Springer, 2001.

10 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: amoebot–a new model for programmable matter. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures (SPAA)*, pages 220–222, 2014.

11 Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. On the runtime of universal coating for programmable matter. In *International Conference on DNA-Based Computers*, pages 148–164. Springer, 2016.

12 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication*, page 21. ACM, 2015.

13 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 289–299. ACM, 2016.

14 David Doty. Theory of algorithmic self-assembly. *Communications of the ACM*, 55:78–88, 2012.

15 David Doty. Timing in chemical reaction networks. In *Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 772–784, 2014.

16 Adrian Dumitrescu and János Pach. Pushing squares around. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 116–123. ACM, 2004.

17 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Formations for fast locomotion of metamorphic robotic systems. *The International Journal of Robotics Research*, 23(6):583–593, 2004.

18 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Motion planning for metamorphic systems: Feasibility, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics and Automation*, 20(3):409–418, 2004.

19 Yuval Emek and Jara Uitto. Dynamic networks of finite state machines. In *International Colloquium on Structural Information and Communication Complexity*, pages 19–34. Springer, 2016.

20 Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by oblivious mobile robots. *Synthesis lectures on distributed computing theory*, 3(2):1–185, 2012.

21 Kyle Gilpin, Ara Knaian, and Daniela Rus. Robot pebbles: One centimeter modules for programmable matter through self-disassembly. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2485–2492. IEEE, 2010.

22 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

23 Ara N. Knaian, Kenneth C. Cheung, Maxim B. Lobovsky, Asa J. Oines, Peter Schmidt-Neilsen, and Neil A. Gershenfeld. The milli-motein: A self-folding chain of programmable matter with a one centimeter module pitch. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1447–1453. IEEE, 2012.

24 Evangelos Kranakis, Danny Krizanc, and Euripides Markou. The mobile agent rendezvous problem in the ring. *Synthesis Lectures on Distributed Computing Theory*, 1(1):1–122, 2010.

25 Yunfeng Lu, Yi Yang, Alan Sellinger, Mengcheng Lu, Jinman Huang, Hongyou Fan, Raid Haddad, Gabriel Lopez, Alan R. Burns, Darryl Y. Sasaki, John Shelnutt, and C. Jeffrey Brinker. Self-assembly of mesoscopically ordered chromatic polydiacetylene/silica nanocomposites. *Nature*, 410(6831):913–917, 2001.

26 Othon Michail and Paul G. Spirakis. Simple and efficient local codes for distributed stable network construction. *Distributed Computing*, 29(3):207–237, 2016. doi:10.1007/s00446-015-0257-4.

27 Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 2017. Accepted: 6th April 2017, To appear.

28 Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, 2006.

29 Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares. In *Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC)*, pages 459–468, 2000. doi:10.1145/335305.335358.

30 Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

31 Masahiro Shibata, Toshiya Mega, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Uniform deployment of mobile agents in asynchronous rings. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 415–424. ACM, 2016.

32 Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, March 1999. doi:10.1137/S009753979628292X.

33 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.

34 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In

Proceedings of the 4th conference on Innovations in Theoretical Computer Science, pages 353–354. ACM, 2013.

35 Yukiko Yamauchi, Taichi Uehara, and Masafumi Yamashita. Brief announcement: pattern formation problem for synchronous mobile robots in the three dimensional euclidean space. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 447–449. ACM, 2016.

36 Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

Distributed Monitoring of Network Properties: The Power of Hybrid Networks*

Robert Gmyr[1], Kristian Hinnenthal[2], Christian Scheideler[3], and Christian Sohler[4]

1 **Paderborn University, Paderborn, Germany**
gmyr@mail.upb.de
2 **Paderborn University, Paderborn, Germany**
krijan@mail.upb.de
3 **Paderborn University, Paderborn, Germany**
scheideler@mail.upb.de
4 **TU Dortmund, Dortmund, Germany**
christian.sohler@tu-dortmund.de

Abstract

We initiate the study of network monitoring algorithms in a class of hybrid networks in which the nodes are connected by an *external network* and an *internal network* (as a short form for externally and internally controlled network). While the external network lies outside of the control of the nodes (or in our case, the monitoring protocol running in them) and might be exposed to continuous changes, the internal network is fully under the control of the nodes. As an example, consider a group of users with mobile devices having access to the cell phone infrastructure. While the network formed by the WiFi connections of the devices is an external network (as its structure is not necessarily under the control of the monitoring protocol), the connections between the devices via the cell phone infrastructure represent an internal network (as it can be controlled by the monitoring protocol). Our goal is to continuously monitor properties of the external network with the help of the internal network. We present scalable distributed algorithms that efficiently monitor the number of edges, the average node degree, the clustering coefficient, the bipartiteness, and the weight of a minimum spanning tree. Their performance bounds demonstrate that monitoring the external network state with the help of an internal network can be done much more efficiently than just using the external network, as is usually done in the literature.

1998 ACM Subject Classification C.2.1 [Network Architecture and Design] Distributed networks, G.2.2 [Graph Theory] Graph Algorithms

Keywords and phrases Network Monitoring, Hybrid Networks, Overlay Networks

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.137

1 Introduction

In this paper we propose a new model for the study of distributed algorithms for communication networks that is based on a class of hybrid networks that is becoming more and more important. In this class of hybrid networks, the nodes are connected by an *external*

* This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901) and within Project A2 of the Collaborative Research Center "Providing Information by Resource-Constrained Analysis" (SFB 876).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 137; pp. 137:1–137:15

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

network and an *internal network*. While the external network is not under the control of the nodes, the internal network is fully under their control. Such hybrid networks can be found at a physical as well as logical level. Consider, for instance, the case that we have a set of wireless devices with access to the cell phone infrastructure that are dispersed over a limited area like a city center so that they can form a connected network using their WiFi connections. The advantage of this type of network is that the devices would in principle be able to exchange information without the use of the cell phone infrastructure, which would save costs. However, this may come at the price of having large message delays and even being unable to handle certain tasks as there might be network partitions from time to time. Therefore, if it is possible to design protocols that only require a small amount of message exchanges via the cell phone infrastructure in order to solve certain tasks much faster and more reliably than via the WiFi network, users may find it acceptable to make use of the cell phone infrastructure. Another example is an expedition or a rescue team that is connected via satellite telephones, which nowadays can support both satellite as well as wireless communication. In the logical world, one can envision a peer-to-peer network formed by friendship links in a social network. Just communicating via these friendship links has the advantage that all interactions are trusted. However, due to the irregular structure of the social network it has the disadvantage that it might be hard to perform certain tasks like network monitoring or finding anyone efficiently. Therefore, it might also be useful to have a network of untrusted links on top of the social network in order to be able to quickly approximate certain properties of it or to find shortest paths. A common theme in all of these examples is having two communication modes that significantly differ concerning their control and in which control comes with costs like financial cost, acceptance, reliability, or integrity. There is already a large body of literature on network algorithms for the case of static or dynamic networks whose topology is not under the control of the nodes. On the other side, there also exists an abundance of network algorithms in which the topology is fully under the control of the nodes, like in peer-to-peer systems. However, to the best of our knowledge, nothing rigorous in the context of network monitoring has been shown yet for *combinations* of these networks, so this paper initiates the rigorous study of this direction.

1.1 Model and Problem Statement

We consider networks with a *static* node set and a *dynamic* edge set. Time proceeds in *synchronous rounds* and for each round i we are given a set of undirected edges E_i. The *external network* in round i is represented by the undirected graph $G_i = (V, E_i)$. We assume that the degree of G_i is polylogarithmic for all i. An algorithm has no control over the edges in E_i, however it can establish additional *overlay edges* to form an *internal network* or *overlay network*: Each node u has a unique *identifier* $\mathrm{id}(u)$ which is a bit string of length $O(\log n)$ where $n = |V|$. Let $D_i(u)$ be the set of identifiers stored by a node u in round i. We define the set of overlay edges in round i as $D_i = \{(u, v) \mid u \in V \text{ and } v \in D_i(u)\}$. A node has immediate access to the identifiers of its neighbors in G_i and can store such an identifier for future reference. In round i, a node u can send a distinct message to each node v such that $\{u, v\} \in E_i$ or $(u, v) \in D_i$. A message sent in round i arrives at the beginning of round $i + 1$. The local memory and computation of the nodes is unbounded. However, a node can send and receive at most polylogarithmically many bits in each round.

We investigate *monitoring problems*. In these problems, a designated node s that we call the *monitor node* or simply *monitor* has to continuously observe a property of the external network like the number of edges or the weight of a minimum spanning tree. Formally, a property p is a function from the set of undirected graphs into some set of property values.

Since the external network G_i is dynamic, the property value $p(G_i)$ can change from round to round. We say an algorithm *monitors a property* p with *setup time* i_0 and *delay* δ if for all rounds $i \geq i_0$ the monitor node outputs the property value $p(G_i)$ by round $i + \delta$. We refer to the first i_0 rounds of the execution of a monitoring algorithm as the *setup phase* and refer to the remaining rounds as the *monitoring phase*. Initially, the set of overlay edges D_0 is empty. An algorithm can use the setup phase to construct an initial internal network that supports the computation of the property value. It can continue to adapt the internal network during the monitoring phase. We assume the graph G_0 to be connected. Beyond this, we make no assumptions about the evolution of the edge set.

1.2 Related Work

In the networking community, the name "hybrid network" has been used in the context of networks containing equipment from multiple vendors, consisting of different physical networks or communication modes, or networks incorporating both peer-to-peer and client-server approaches. These topics are not related to our work, so we do not consider them.

There is a large body of literature on overlay networks, especially in the context of peer-to-peer systems. Whereas most of the proposed overlay networks do not worry about the underlying network, there is also a number of proposals for so-called locality-aware overlays, with prominent examples like Tapestry [37] and Pastry [34]. However, these constructions are only concerned about adapting or optimizing the overlay to the underlying network and do not aim at monitoring properties of the underlying network with the help of the overlay.

The dynamic external network assumed by our model is related to the *dynamic graph model* introduced by Kuhn et al. [24], in which an adversary changes the edge set of a graph in every round. Kuhn et al. [24] focus on solving the counting and the token dissemination problem in that model, which has been further considered, for example, in [12, 15] (see [5] for an overview). Abshoff and Meyer auf der Heide study how to perform continuous aggregation in these networks [1]. However, like the other works in this area, they do not consider establishing additional overlay edges. Another approach related to ours is the work by Michail and Spirakis [28], which extends the population protocol model to a model in which nodes can decide whether to keep connections proposed to them or not, but there is no underlying network to monitor.

Some of our algorithms make use of techniques particularly known from the field of parallel computation. For example, it is well-known how to use *pointer jumping* [18] in order to perform rapid tree traversals in PRAMs (see e.g. [3, 19, 35]). Furthermore, there exists an abundance of parallel algorithms computing MSTs in such models, the best of which achieve a runtime of $O(\log n)$ (see [16] for an overview). The algorithm presented in Section 2 has some similarities with [19]. However, we are not aware of any distributed implementation of such an algorithm with runtime $o(\log^2 n)$ that does not cause high node congestion. This is also the problem with the various algorithms proposed for the congested clique model, which has recently received a considerable amount of attention (e.g., [7, 11, 17, 25, 27]).

Our algorithms make extensive use of a subroutine for the construction of overlay networks that we present in Section 2. This subroutine transforms a given graph into a rooted tree of constant degree and depth $O(\log n)$. Angluin et al. [2] proposed a similar subroutine that achieves the same result and that even works in an asynchronous setting. However, the subroutine of Angluin et al. is randomized while our subroutine is deterministic. As a consequence, all monitoring algorithms presented in this work are also fully deterministic. Furthermore, our subroutine can be used for the efficient construction of a spanning tree of a given graph, which cannot directly be achieved using the approach by Angluin et al.

Table 1 This table summarizes the results of this work. W is the maximum weight of an edge in the graph. The algorithm for monitoring the exact weight of a minimum spanning tree requires integral edge weights while the approximation algorithm has no such requirement. The latter algorithm approximates the weight M up to an additive factor of $\pm\varepsilon M$.

Monitoring Problem	Setup Time	Delay	Section
Number of Edges	$O(\log^2 n)$	$O(\log n/\log\log n)$	3
Average Node Degree	$O(\log^2 n)$	$O(\log n/\log\log n)$	3
Clustering Coefficient	$O(\log^2 n)$	$O(\log n/\log\log n)$	3
Bipartiteness	0	$O(\log^2 n)$	4
Exact MST Weight	0	$O(W+\log^2 n)$	5.1
Approximate MST Weight	$O(\log^2 n)$	$O(\log(W)/\varepsilon\cdot\log^2(W/\varepsilon)+\log n/\log\log n)$	5.2

In the algorithms presented in Section 3 the monitor continuously collects data from the nodes of the network by performing *aggregation*. There is a huge amount of work on aggregation in the context of sensor networks, but research in this area has focused on monitoring environmental properties, the state of systems (like bridges or airplanes) or facilities (like warehouses). Distributed aggregation has also been studied extensively for conventional, static networks (see, e.g., [4, 22, 23] or [26] for a comprehensive overview), but not for hybrid forms as considered in this paper.

One of the network properties considered in this work is the weight of a minimum spanning tree (or MST), see Section 5. The problem of computing an MST in a distributed manner is well studied, see for example [13, 14, 32, 33]. The problem of computing only the *weight* of an MST instead of the MST itself has been studied in the area of sequential sublinear algorithms. This line of research was initiated by Chazelle et al. [8] and continued in [6, 9, 10]. Our algorithms for monitoring the weight of an MST apply the ideas of Chazelle et al. [8] in a distributed context and also incorporate some ideas from [10].

1.3 Our Contribution

We initiate the study of hybrid networks consisting of externally and internally controlled edges and present *deterministic* algorithms for *monitoring network properties* in such networks. Our results are summarized in Table 1. As a byproduct of the algorithms for monitoring the weight of a minimum spanning tree, we also present algorithms for the distributed computation of minimum spanning trees in hybrid networks. Since the delays trivially increase to $\Omega(n)$ in the worst case when just using an external network, our results demonstrate that with the help of hybrid networks monitoring can be done exponentially faster compared to just having an external network.

2 Setup Phase

All monitoring algorithms presented in this work that rely on a dedicated setup phase use a common algorithm for the construction of the initial overlay network. This algorithm organizes the nodes into a tree T of polylogarithmic degree and depth $O(\log n/\log\log n)$ that is rooted at the monitor. In this section, we first present a more general algorithm that we refer to as the *Overlay Construction Algorithm*. This algorithm shares some similarities with an algorithm of Angluin et al. [2]. The algorithm is also frequently used as a subroutine throughout the remainder of this work. Based on this algorithm, we describe how the desired

tree T can be constructed at the end of the section. For simplicity, we assume that every node knows the total number of nodes n. The algorithms can be modified to remove this assumption.

For a given bidirected connected graph G of polylogarithmic degree, the Overlay Construction Algorithm arranges the nodes of G into a tree of constant degree and depth $O(\log n)$. On a high level, the algorithm works as follows. It operates on *supernodes* which are groups of nodes that act in coordination. Let the identifier of a supernode be the highest identifier of the nodes it contains. Define two supernodes u, v to be adjacent if there are nodes x, y that are adjacent in G such that x is in u and y is in v. Initially, each node forms a supernode on its own. The algorithm alternatingly executes a *grouping step* and a *merging step*. In the grouping step, each supernode u determines the neighboring supernode v with the highest identifier. If $\mathrm{id}(v) > \mathrm{id}(u)$ then u sends a *merge request* to v. Consider the graph whose node set is the set of all supernodes and that contains a directed edge (u, v) if u sent a merge request to v. Since each supernode sends at most one merge request to a supernode of higher identifier, this graph is a forest. During the merging step, each tree of this forest is merged into a new supernode. Before we describe how this high-level algorithm can be implemented by the nodes, we analyze the number of iterations of consecutive grouping and merging steps until only a single supernode remains. The following lemma can be shown by observing that each supernode merges with another supernode within at most 2 iterations.

▶ **Lemma 1.** *After $O(\log n)$ iterations only a single supernode remains.*

At the beginning of every *grouping step*, the following *invariant* holds: Each supernode is internally organized in an overlay forming a tree of constant degree and depth $O(\log n)$ that is rooted at the node with the highest identifier and each node knows the identifier of its supernode. The nodes cooperatively simulate the behavior of their respective supernodes during the grouping step as follows. Consider a supernode u and the corresponding internal tree T_u. First, every node of u sends $\mathrm{id}(u)$ along every incident edge in the original graph G. Thereby, every node learns the identifiers of its neighboring supernodes. Then, the nodes of u use a convergecast along T_u to determine the identifier of the supernode v with the highest identifier among the neighbors of u. This convergecast also collects the identifier of the node x with the highest identifier in u that is adjacent to a node in v. Once this convergecast is complete, the root of T_u knows both $\mathrm{id}(v)$ and $\mathrm{id}(x)$. If $\mathrm{id}(v) > \mathrm{id}(u)$ then the root of u sends a message to x. Upon receiving this message, x sends a merge request to a neighboring node in v and sends a broadcast through T_u to establish itself as the new root of T_u. The nodes in G wait with starting the merging step until $O(\log n)$ rounds have passed to guarantee that the above operations are completed in all supernodes and all nodes start the merging step at the same time.

At the beginning of every *merging step*, we have the following situation. Consider the graph consisting of the internal trees of all supernodes together with all edges along which a merge request has been sent. This graph is a forest and the trees of this forest form the new supernodes resulting from the merging step. Therefore, the nodes of each new supernode v are already arranged into a tree T_v. Furthermore, v contains exactly one former root node that did not instruct a node to send a merge request. It is not hard to see that this node has the highest identifier in v and therefore becomes the root of T_v. In its current state, T_v can have up to polylogarithmic degree and linear depth. To restore the invariants required at the beginning of a grouping step, we have to transform T_v into a tree of constant degree and depth $O(\log n)$. Furthermore, we have to make sure that all nodes in v know the identifier $\mathrm{id}(v)$ of the root of T_v.

First, we reorganize T_v into a *child-sibling tree*. For this, each inner node y arranges its children into a path sorted by increasing identifier and only keeps the child with the lowest identifier. Each former child of y changes its parent to be its predecessor on the path and stores its successor as a *sibling*. In the resulting child-sibling tree, each node stores at most three identifiers: a parent, a sibling, and a child. By interpreting the sibling of a node as a second child, we get a binary tree. This transformation of T_v into a binary tree takes $O(1)$ rounds.

Based on this binary tree, we construct a *ring* of *virtual nodes* as follows. Consider the depth-first traversal of the tree that visits the children of each node in order of increasing identifier. A node occurs at most three times in this traversal. Let each node act as a distinct virtual node for each such occurrence and let $k \leq 3n$ be the number of virtual nodes. A node can locally determine the predecessor and successor of its virtual nodes according to the traversal. Therefore, the nodes can connect their virtual nodes into a ring in $O(1)$ rounds.

Next, we use pointer jumping to quickly add *chords* (i.e., shortcut edges) to the ring. The virtual nodes execute the following protocol for $\lfloor \log n \rfloor + 1 \geq \lfloor \log k \rfloor - 1$ rounds. Each virtual node y learns two identifiers ℓ_t and r_t in each round t of this protocol. Let ℓ_0 and r_0 be the predecessor and successor of y in the ring. In round t, y sends ℓ_t to r_t and vice versa. At the beginning of round $t+1$, y receives one identifier from ℓ_t and r_t, respectively. It sets ℓ_{t+1} to the identifier received from ℓ_t and r_{t+1} to the identifier received from r_t. It then proceeds to the next round of the protocol. In every round of this protocol each virtual node adds a new chord to the ring by introducing its latest neighbors to each other. The distance between these neighbors w.r.t the ring doubles from round to round up to the point where the distance exceeds the number of virtual nodes k. Based on this observation, it is not hard to show that after the specified number of rounds, the diameter of the graph has reduced to $O(\log n)$ while the degree has grown to $O(\log n)$. Once the protocol finished, the root of v initiates a broadcast from one of its virtual nodes followed by a convergecast to determine the number of virtual nodes k.

Finally, we use the chords to construct a binary tree of depth $O(\log n)$. For this, the root of v initiates a broadcast by sending a message to its neighbors $\ell_{t'}$ and $r_{t'}$ where $t' = \lfloor \log k \rfloor - 1$. A node that receives the broadcast after t steps forwards it to $\ell_{t'}$ and $r_{t'}$ where $t' = \max\{\lfloor \log k \rfloor - t - 1, 0\}$. It is not hard to see that the binary tree constructed by this broadcast has depth $O(\log n)$ and contains all nodes of the ring. At this point, the nodes discard all overlay edges constructed so far and only keep the edges of the binary tree. We then merge the virtual nodes back together such that each node adopts the edges of its virtual nodes. This results in a graph of degree at most 6 and diameter $O(\log n)$. Note that this graph is not necessarily a tree. To construct a tree that satisfies the invariants for the grouping step, the root of v sends another broadcast through the resulting graph to construct a breadth-first search tree that has constant degree and diameter $O(\log n)$. This broadcast also informs all nodes in v about $\mathrm{id}(v)$. The operations described above take $O(\log n)$ rounds overall. As before, all nodes in G wait for $O(\log n)$ rounds to pass so that they enter the next grouping step at the same time.

Once only a single supernode u remains, which is the case if during a grouping step no node reports the identifier of a neighboring supernode, T_u covers all nodes of G and has the desired properties. Since the algorithm runs for $O(\log n)$ iterations and each iteration takes $O(\log n)$ rounds, we have the following theorem.

▶ **Theorem 2.** *Given any bidirected connected graph G of n nodes and polylogarithmic degree, the Overlay Construction Algorithm constructs a constant degree tree of depth $O(\log n)$ that contains all nodes of G and that is rooted at the node with the highest identifier. The algorithm takes $O(\log^2 n)$ rounds.*

Theorem 2 directly implies the following corollary.

▶ **Corollary 3.** *Consider a bidirected graph G of n nodes and polylogarithmic degree. For each connected component C of G, the Overlay Construction Algorithm constructs a constant degree tree of depth $O(\log |C|)$ that contains all nodes of C and that is rooted at the node with the highest identifier in C. The algorithm takes $O(\log^2 |C|)$ rounds in each component and $O(\log^2 n)$ rounds overall.*

Finally, note that the algorithm only sends merge requests along edges of G. This gives rise to the following observation.

▶ **Observation 4.** *For a bidirected connected graph G, the edges along which the Overlay Construction Algorithm sends the merge requests form a spanning tree of G.*

So by letting the nodes locally mark the edges that carry a merge request, the algorithm can be used for the distributed construction of a spanning tree of G in $O(\log^2 n)$ time. While this observation is not immediately relevant for the setup phase, it will be useful in later sections.

Based on the Overlay Construction Algorithm, it is easy to achieve the goal for the setup phase of organizing the nodes into a tree T of polylogarithmic degree and depth $O(\log n / \log\log n)$ that is rooted at the monitor s. At the beginning of the setup phase, each node stores the identifiers of its neighbors in the given graph G_0 that represents the external network. This effectively creates a bidirected overlay network that directly corresponds to the undirected graph G_0. Note that G_0 is connected by assumption. Therefore, we can use the Overlay Construction Algorithm to construct a tree of constant degree and depth $O(\log n)$ that contains all nodes in the network. Once the algorithm terminates, s broadcasts a message through the resulting tree to establish itself as the new root. This does not increase the asymptotic depth of the tree. We then decrease the depth to $O(\log n / \log\log n)$ as follows. Each node x broadcasts its identifier down the tree up to a distance of $\lceil \log\log n \rceil$. Every node that receives the broadcast of x establishes an edge to x. It is not hard to see that this creates a graph of at most polylogarithmic degree and diameter $O(\log n / \log\log n)$. Finally, s sends a broadcast through this graph to create a breadth-first search tree that has the desired properties. We have the following theorem.

▶ **Theorem 5.** *A setup time of $O(\log^2 n)$ rounds is sufficient to organize the nodes of the network into a tree T of polylogarithmic degree and depth $O(\log n / \log\log n)$.*

Unless otherwise stated, we assume in the following sections that the setup phase is executed as described above.

3 Three Simple Monitoring Problems

In order to introduce some basic concepts that underlie all monitoring algorithms presented in this work, we first consider three simple monitoring problems. Specifically, we show how to monitor the number of edges, the average node degree, and the clustering coefficient of the network by performing *aggregation* on the tree T constructed during the setup phase.

Consider the problem of monitoring the number of edges. We first present an algorithm that efficiently determines the number of edges in a graph G and then show how this algorithm can be used to continuously monitor the number of edges. It is well known that the number of edges in a graph is $|E| = 1/2 \cdot \sum_{u \in V} \deg(u)$ where $\deg(u)$ is the degree of a node u. Therefore, we can compute $|E|$ by aggregating the sum of all node degrees in the following way. In the first round, each leaf node u in T sends $\deg(u)$ to its parent. Once an inner node

u has received a value x_j from each of its children, it sends $\deg(u) + \sum_j x_j$ to its parent. After $O(\log n / \log\log n)$ rounds, the monitor s has received a value from each of its children and can use these values together with its own degree to compute $|E|$ as described above.

To continuously monitor $|E_i|$ for every $i \geq i_0$, the above algorithm is executed in a *pipelined fashion*: In each round $i \geq i_0$ a new instance of the algorithm is started. The instances run in parallel and do not interact with each other. At the beginning of a round i, each node stores the identifiers of its neighbors in the graph G_i that represents the external network to create a copy of the graph in form of an overlay network that the algorithm can operate on. This copy is discarded once the corresponding instance of the algorithm terminates. The messages sent by an instance of the algorithm are labeled with the round number in which the instance was started so that received messages can be correctly assigned. Note that the number of bits a node sends and receives per round in the given algorithm is polylogarithmic. Since the delay of the algorithm is also polylogarithmic, the number of bits a node sends and receives in the pipelined execution is polylogarithmic as well. We have the following theorem.

▶ **Theorem 6.** *The number of edges can be monitored with setup time $O(\log^2 n)$ and delay $O(\log n / \log\log n)$.*

In the remainder of this work, we only present algorithms that compute the value of a network property for a single graph G and implicitly assume that the respective algorithm is executed in a pipelined fashion to solve the monitoring problem under consideration.

Based on the ideas of the algorithm above, it is easy to solve a number of monitoring problems that can be reduced to aggregation. For example, one can monitor the average node degree by letting s multiply the result of the given algorithm with $2/n$ before outputting it. This gives us the following corollary.

▶ **Corollary 7.** *The average node degree can be monitored with setup time $O(\log^2 n)$ and delay $O(\log n / \log\log n)$.*

As a final example, we consider the *clustering coefficient* of a network [36]. Intuitively, the clustering coefficient reflects the relative number of triangles in a graph G. It is particularly relevant in the context of biological and social networks [29, 30, 31, 36]. Formally, the clustering coefficient of a node u is defined as

$$C(u) = \frac{2 \cdot |\{v, w \in N(u) \mid \{v, w\} \in E\}|}{\deg(u) \cdot (\deg(u) - 1)},$$

where $N(u)$ is the set of neighbors of u. The clustering coefficient of a graph G is defined as $C(G) = 1/n \cdot \sum_{u \in V} C(u)$. Each node u can compute $C(u)$ in constant time by communicating with its neighbors. Therefore, $C(G)$ can be computed by aggregating the sum of all $C(u)$ along T and dividing the result by n at the monitor. We have the following theorem.

▶ **Theorem 8.** *The clustering coefficient can be monitored with setup time $O(\log^2 n)$ and delay $O(\log n / \log\log n)$.*

4 Bipartiteness

In this section, we consider the problem of monitoring whether the network forms a *bipartite* graph. Our algorithm is based on the following commonly known approach (see e.g. [20]). Given a connected graph G, compute a rooted spanning tree of G and assign a color from $\{0, 1\}$ to each node that corresponds to the parity of its depth in the spanning tree. We say

an edge is *valid* if it connects nodes of different colors and *invalid* otherwise. G is bipartite if and only if all edges are valid. It remains to show how this approach can be implemented efficiently in our framework.

According to Observation 4, we can use the Overlay Construction Algorithm to mark the edges of a spanning tree S in G. Define the monitor s to be the root of S. Each node has to determine the parity of its depth in S to set its color. Since S might have linear depth, a simple broadcast from s does not constitute an efficient solution to this problem. Instead, we use pointer jumping along a depth-first traversal of S to determine the colors of the nodes. We define the traversal of S as follows. The traversal starts at s and moves to the neighbor of s in S with the lowest identifier. For a node u, let $u_0, \ldots, u_{\deg(u)-1}$ be the neighbors of u in S arranged by increasing identifier. When the traversal reaches u from a node u_i, it continues on to node $u_{(i+1) \bmod \deg(u)}$. The traversal finishes when it reaches s from the neighbor of s in S with the highest identifier. Define the *traversal distance* $d(u)$ to be the number of steps required to reach u for the first time in this traversal. As we will show in Lemma 9, the parity of $d(u)$ equals the parity of the depth of u in S. For now, we focus on computing $d(u)$ efficiently for all nodes.

Each node u simulates one virtual node for each occurrence of u in the traversal, and the nodes connect these virtual nodes into a ring. More specifically, each node u simulates virtual nodes $v_0, \ldots, v_{\deg(u)-1}$ such that v_i is the successor of a virtual node of u_i and the predecessor of a virtual node of $u_{(i+1) \bmod \deg(u)}$ in the ring. Note that the resulting ring consists of $2(n-1)$ virtual nodes. We use pointer jumping to add chords to the ring following the same protocol we used during the merging step of the Overlay Construction Algorithm. We define ℓ_0 to be the successor of a virtual node and r_0 to be the predecessor of a virtual node. We execute the protocol for $t = \lfloor \log(2(n-1)) \rfloor$ rounds so that each node constructs chords ℓ_i and r_i for each $1 \leq i \leq t$. Each chord ℓ_i (resp. r_i) bridges a distance of exactly 2^i along the ring. The chords allow us to efficiently compute the values $d(u)$ in the following way. Let v^* be the virtual node simulated by s that precedes a virtual node of the neighbor of s with the lowest identifier. v^* stores the value 0 and initiates a broadcast by sending a message with value 2^i along each chord ℓ_i for $0 \leq i \leq t$. Consider a virtual node that receives a broadcast message and that has not yet stored a value. Let x be the value associated with the received message. The virtual node stores x and sends a message containing the value $x + 2^i$ along each chord ℓ_i for $0 \leq i \leq t$. It is not hard to see that this broadcast reaches all virtual nodes within $O(\log n)$ rounds and the value stored at a virtual node v after the broadcast finishes corresponds to the length of the path from v^* to v along the ring. Therefore, each node u can determine the value $d(u)$ by taking the minimum of the values stored at its virtual nodes. Once all nodes computed their traversal distance in this way, each node u determines whether it is incident to an invalid edge by checking for each neighbor w in G whether $d(u) \equiv d(w) \mod 2$. Then, the nodes use a convergecast to inform s whether there is an invalid edge. If so, s outputs that G is not bipartite. Otherwise, s outputs that G is bipartite.

To establish the correctness of the algorithm, we show the following lemma.

▶ **Lemma 9.** *For each node u, the parity of the depth of u equals the parity of $d(u)$.*

Proof. Note that the tree is traversed in a depth-first order. Therefore, the traversal takes an even number of steps between any two visits of the same node. Let P be the shortest path from s to u in S. The length of P equals the depth of u. The traversal follows P but takes a detour whenever it explores a branch outside of P. By the argument above, each such detour has even length. Therefore, the parity of the depth of u equals the parity of $d(u)$. ◀

We can monitor bipartiteness for a disconnected graph by performing the above algorithm on its connected components and aggregating the results on the tree T built in the setup phase. Furthermore, we can reduce the setup time to 0 by delaying the first monitoring phase until T is constructed, which does not asymptotically increase the total delay. This implies the following theorem.

▶ **Theorem 10.** *The bipartiteness of a graph can be monitored with setup time* 0 *and delay* $O(\log^2 n)$.

5 Minimum Spanning Tree

We now turn to the problem of monitoring the *weight* of a *minimum spanning tree* (or *MST*). We assume that the edge set changes from round to round but the external network always stays connected. Additionally, we assume that each edge has a weight that can also change every round. We present an algorithm that monitors the exact MST weight in Section 5.1 and an algorithm that monitors an approximation of the MST weight with a shorter delay in Section 5.2. Both algorithms are based on a sequential approximation algorithm by Chazelle et al. [8]. As a byproduct, we describe in Section 5.3 how the algorithms for computing the MST weight can be adapted for the distributed computation of an actual MST.

5.1 Exact MST Weight

The main idea behind the algorithm is to reduce the computation of the weight of an MST in a graph G to counting the number of connected components in certain subgraphs of G. This idea was first introduced by Chazelle et al. [8]. We assume that the edge weights are taken from the set $\{1, 2, \dots, W\}$ for some given $W \in \mathbb{N}$. Define the *threshold graph* $G^{(\ell)}$ to be the subgraph of G consisting of all edges with weight at most ℓ, and define $c^{(\ell)}$ to be the number of connected components in $G^{(\ell)}$. The MST weight M can be computed from the values $c^{(\ell)}$ as shown in the following lemma.

▶ **Lemma 11** (Chazelle et al. [8])**.** *In a graph with edge weights from* $\{1, 2, \dots, W\}$*, the MST weight is* $M = n - W + \sum_{i=1}^{W-1} c^{(i)}$.

Based on Lemma 11, the monitor can compute the MST weight as follows. Consider the threshold graph $G^{(\ell)}$ for some $\ell \in \{1, 2, \dots W-1\}$. According to Corollary 3, executing the Overlay Construction Algorithm on $G^{(\ell)}$ creates an overlay network in which each connected component of $G^{(\ell)}$ is spanned by a rooted tree of overlay edges. Each node knows whether it is a root of one of these trees. Therefore, we can determine $c^{(\ell)}$ by counting the number of roots, which can easily be achieved using aggregation along the tree T that results from the setup phase. By iterating this process, the monitor learns the value $c^{(\ell)}$ for each $\ell \in \{1, 2, \dots W-1\}$. It can then use the equation given in Lemma 11 to compute the MST weight. Since T is only used after the algorithm already ran for $O(\log^2 n)$ rounds, we can construct T during the monitoring phase and therefore skip the setup phase. Furthermore, we can reduce the delay by computing up to $\log^2 n$ different $c^{(i)}$'s in parallel. This implies the following theorem.

▶ **Theorem 12.** *For edge weights from* $\{1, 2, \dots, W\}$*, the MST weight can be monitored with setup time* 0 *and delay* $O(W + \log^2 n)$.

5.2 Approximate MST Weight

Next, we present an algorithm that monitors the MST weight with a significantly shorter delay at the cost of a small approximation error. The algorithm is less restrictive in that it allows the edge weights to be real numbers from the interval $[1, W]$ for a given $W \in \mathbb{R}$. It is based on the same general idea as the algorithm from the previous section but additionally incorporates ideas from the work of Czumaj and Sohler [10].

First, each node rounds up the edge weight of each incident edge to a power of $(1+\varepsilon)$ for a fixed ε with $0 < \varepsilon \leq 1$. In the resulting graph G', each edge weight is of the form $(1+\varepsilon)^i$ where $0 \leq i \leq \log_{1+\varepsilon} W$. Let M' be the MST weight in G'. We have the following lemma, which is analogous to Lemma 11 from the previous section.

▶ **Lemma 13** (Czumaj and Sohler [10]). *In a graph with edge weights of the form $(1+\varepsilon)^i$ for $0 \leq i \leq \log_{1+\varepsilon} W$, the MST weight is $M' = n - W + \varepsilon \cdot \sum_{i=0}^{\log_{1+\varepsilon} W - 1} (1+\varepsilon)^i \cdot c^{((1+\varepsilon)^i)}$.*

Based on Lemma 13, we can compute M' by determining the number of connected components $c^{((1+\varepsilon)^i)}$ in $\log_{1+\varepsilon} W$ many threshold graphs. While this already implies an improvement over the algorithm from the previous section, we can further reduce the delay by ignoring large components in the threshold graphs.

Consider some threshold graph $G^{((1+\varepsilon)^i)}$. We execute the Overlay Construction Algorithm as in the previous section but we stop its execution after $O(\log^2(2W/\varepsilon))$ rounds. By Corollary 3, the algorithm is guaranteed to finish its computation in each connected component of size at most $2W/\varepsilon$. In larger connected components, the algorithm may finish but is not guaranteed to do so. It is easy to modify the algorithm such that all nodes of a connected component know whether the algorithm finished its computation for that connected component. This allows us to ignore root nodes in connected components for which the algorithm did not finish. Thereby, the algorithm establishes a unique root node for each connected component of size at most $2W/\varepsilon$ while in each larger connected component either a unique root node is established or no root is established. Let $\hat{c}^{((1+\varepsilon)^i)}$ be the number of root nodes established in this way. The nodes determine the value of $\hat{c}^{((1+\varepsilon)^i)}$ using aggregation along the tree T constructed in the setup phase.

As in the previous section, the nodes iteratively execute this process for each i such that $0 \leq i \leq \log_{1+\varepsilon} W$. After the Overlay Construction Algorithm finishes for an iteration, the nodes start the aggregation for counting the number of roots. The nodes do not wait for this aggregation to finish but rather execute it in parallel to the next iteration. Thereby, we slightly interleave consecutive iterations which reduces the overall delay. After the monitor has learned the values $\hat{c}^{((1+\varepsilon)^i)}$, it computes and outputs $\hat{M} = n - W + \varepsilon \cdot \sum_{i=0}^{\log_{1+\varepsilon} W - 1} (1+\varepsilon)^i \cdot \hat{c}^{((1+\varepsilon)^i)}$. We have the following theorem.

▶ **Theorem 14.** *For edge weights from $[1, W]$, the MST weight M can be monitored up to an additive term of $\pm\varepsilon M$ for any $0 < \varepsilon \leq 1$ with setup time $O(\log^2 n)$ and delay $O\left(\frac{\log W}{\varepsilon} \cdot \log^2\left(\frac{W}{\varepsilon}\right) + \frac{\log n}{\log\log n}\right)$.*

Proof. We first show the approximation factor. Rounding up the edge weights to a power of $(1+\varepsilon)$ increases the MST weight by a factor of at most $(1+\varepsilon)$. Therefore, we have $M \leq M' \leq (1+\varepsilon) \cdot M$. When computing the values $\hat{c}^{((1+\varepsilon)^i)}$ the algorithm potentially ignores all connected components of size larger than $2W/\varepsilon$. In each threshold graph there are at most $\varepsilon n/(2W)$ such connected components. The algorithm cannot overestimate the number of connected components. Therefore, we have $c^{((1+\varepsilon)^i)} - \varepsilon n/(2W) \leq \hat{c}^{((1+\varepsilon)^i)} \leq c^{((1+\varepsilon)^i)}$. For the upper bound on the output $\hat{M}$ of the algorithm, the equations above together with the definitions of M' and $\hat{M}$ directly imply $\hat{M} \leq M' \leq (1+\varepsilon) \cdot M$. For the lower bound on $\hat{M}$, we

have $\hat{M} \geq n - W + \varepsilon \cdot \sum_{i=0}^{\log_{1+\varepsilon} W - 1} (1+\varepsilon)^i \cdot \left(c^{((1+\varepsilon)^i)} - \frac{\varepsilon n}{2W} \right) = M' - \frac{\varepsilon^2 n}{2W} \cdot \sum_{i=0}^{\log_{1+\varepsilon} W - 1} (1+\varepsilon)^i \geq M' - \frac{\varepsilon}{2} \cdot n \geq (1-\varepsilon) \cdot M$, where we assume $n \geq 2$ so that $n \leq M + 1 \leq 2M$ for the last inequality.

We now turn to the delay of the algorithm. The algorithm iteratively computes the values $\hat{c}^{((1+\varepsilon)^i)}$ for $\log_{1+\varepsilon} W = O(\log(W)/\varepsilon)$ threshold graphs. In each of these iterations the modified Overlay Construction Algorithm is executed for $O(\log^2(W/\varepsilon))$ rounds. After the Overlay Construction Algorithm finishes in the last iteration, the nodes have to wait for the final aggregation to complete. This takes an additional $O(\log / \log \log n)$ rounds. ◀

Finally, if $W = n^{O(1)}$ we can execute $\log W$ iterations of the algorithm in parallel which gives us the following corollary.

▶ **Corollary 15.** *For edge weights from $[1, W]$ where $W = n^{O(1)}$, the MST weight M can be monitored up to an additive term of $\pm\varepsilon M$ for any $0 < \varepsilon \leq 1$ with setup time $O(\log^2 n)$ and delay $O\left(\frac{1}{\varepsilon} \cdot \log^2\left(\frac{W}{\varepsilon}\right) + \frac{\log n}{\log \log n}\right)$.*

5.3 Distributed Computation of MSTs

Based on the ideas of the previous two sections, we can devise algorithms that mark the edges of an MST instead of just computing the MST weight. While these algorithms do not fit into the monitoring context, they represent natural extensions of the given algorithms and demonstrate that the underlying ideas might be useful outside of network monitoring.

First, we run the Overlay Construction Algorithm on the graph $G^{(1)}$ and mark all edges along which a merge request is sent. We then add the edges of weight 2 and run the Overlay Construction Algorithm again to further merge the supernodes while still marking edges as before. We keep adding edges of increasing weight in this way until we reach the threshold graph $G^{(W)}$ in which only a single supernode remains. By Observation 4, the marked edges form a spanning tree of G. Furthermore, at any given time the algorithm only adds edges of minimal weight to the spanning tree. Therefore, the algorithm is essentially a distributed implementation of Kruskal's Algorithm [21] so that the spanning tree computed by the algorithm is in fact an MST. We have the following theorem.

▶ **Theorem 16.** *Consider a network that initially forms a bidirected connected graph G with edge weights from $\{1, 2, \ldots, W\}$ for some $W \in \mathbb{N}$. There is an algorithm that marks the edges of an MST in G in $O(W \log^2 n)$ rounds.*

Combining this approach with the ideas from Section 5.2 gives us the following theorem.

▶ **Theorem 17.** *Consider a network that initially forms a bidirected connected graph G with edge weights from $[1, W]$ for some $W \in \mathbb{R}$. Let M be the weight of an MST in G. There is an algorithm that marks the edges of a spanning tree in G with weight M' such that $M \leq M' \leq (1+\varepsilon) \cdot M$ in $O(1/\varepsilon \cdot \log W \cdot \log^2 n)$ rounds.*

6 Future Work

We were only able to present a small number of monitoring problems in hybrid networks in this work. However, there is an abundance of classical problems in the literature that can be newly investigated under this model. As we tried to demonstrate in Section 5.3, the idea of using the ability to establish new edges to speed up computation can also be applied outside of network monitoring. We would be very interested in seeing further applications of this idea in other contexts.

References

1 Sebastian Abshoff and Friedhelm Meyer auf der Heide. Continuous aggregation in dynamic ad-hoc networks. In Magnús M. Halldórsson, editor, *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, volume 8576 of *Lecture Notes in Computer Science*, pages 194–209. Springer, 2014. doi:10.1007/978-3-319-09620-9_16.

2 Dana Angluin, James Aspnes, Jiang Chen, Yinghua Wu, and Yitong Yin. Fast construction of overlay networks. In *Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA'05, page 145, 2005. doi:10.1145/1073970.1073991.

3 Mikhail Atallah and Uzi Vishkin. Finding euler tours in parallel. *Journal of Computer and System Sciences*, 29(3):330–337, 1984. doi:10.1016/0022-0000(84)90003-5.

4 Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. John Wiley and Sons, Inc., 2nd edition, 2004.

5 John Augustine, Gopal Pandurangan, and Peter Robinson. Distributed algorithmic foundations of dynamic networks. *SIGACT News*, 47(1):69–98, 2016. doi:10.1145/2902945.2902959.

6 Petra Berenbrink, Bruce Krayenhoff, and Frederik Mallmann-Trenn. Estimating the number of connected components in sublinear time. *Information Processing Letter*, 114(11):639–642, 2014. doi:10.1016/j.ipl.2014.05.008.

7 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC'15, pages 143–152, 2015. doi:10.1145/2767386.2767414.

8 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005.

9 Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM J. Comput.*, 35(1):91–109, 2005. doi:10.1137/S0097539703435297.

10 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM J. Comput.*, 39(3):904–922, 2009. doi:10.1137/060672121.

11 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC'14, pages 367–376, 2014. doi:10.1145/2611462.2611493.

12 Chinmoy Dutta, Gopal Pandurangan, Rajmohan Rajaraman, Zhifeng Sun, and Emanuele Viola. On the complexity of information spreading in dynamic networks. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 717–736. SIAM, 2013. doi:10.1137/1.9781611973105.52.

13 Michael Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, STOC'04, pages 331–340, 2004. doi:10.1145/1007352.1007407.

14 Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. *Journal of Computer and System Sciences*, 72(8):1282–1308, 2006. doi:10.1016/j.jcss.2006.07.002.

15 Bernhard Haeupler and David Karger. Faster information dissemination in dynamic networks via network coding. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing*, PODC'11, pages 381–390, 2011. doi:10.1145/1993806.1993885.

16 Shay Halperin and Uri Zwick. Optimal randomized EREW PRAM algorithms for finding spanning forests. *Journal of Algorithms*, 39(1):1–46, 2001. doi:10.1006/jagm.2000.1146.

17 James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and mst. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC'15, pages 91–100, 2015. doi:10.1145/2767386.2767434.

18 Joseph JaJa. *An Introduction to Parallel Algorithms*, volume 17. Addison Wesley, 1992.

19 Donald B. Johnson and Panagiotis Metaxas. A parallel algorithm for computing minimum spanning trees. *Journal of Algorithms*, 19(3):383–401, 1995. doi:10.1006/jagm.1995.1043.

20 Jon Kleinberg and Eva Tardos. *Algorithm Design: Pearson New International Edition*. Pearson Education Limited, 2013.

21 Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. doi:10.2307/2033241.

22 Fabian Kuhn, Thomas Locher, and Stefan Schmid. Distributed computation of the mode. In *Proceedings of the 27th ACM Symposium on Principles of Distributed Computing*, PODC'08, pages 15–24, 2008. doi:10.1145/1400751.1400756.

23 Fabian Kuhn, Thomas Locher, and Roger Wattenhofer. Tight bounds for distributed selection. In *Proceedings of the 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA'07, pages 145–153, 2007. doi:10.1145/1248377.1248401.

24 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, STOC'10, pages 513–522, 2010. doi:10.1145/1806689.1806760.

25 Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *Symposium on Principles of Distributed Computing*, PODC'13, pages 42–50, 2013. doi:10.1145/2484239.2501983.

26 Thomas Locher. *Foundations of aggregation and synchronization in distributed systems*. PhD thesis, ETH Zürich, 2009. doi:10.3929/ethz-a-005799819.

27 Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in O(log log n) communication rounds. *SIAM J. Comput.*, 35(1):120–131, 2005. doi:10.1137/S0097539704441848.

28 Othon Michail and Paul G. Spirakis. Simple and efficient local codes for distributed stable network construction. In *ACM Symposium on Principles of Distributed Computing*, PODC'14, pages 76–85, 2014. doi:10.1145/2611462.2611466.

29 Mark E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001. doi:10.1073/pnas.98.2.404.

30 Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):26118, jul 2001. doi:10.1103/PhysRevE.64.026118.

31 Mark E. J. Newman, Duncan J. Watts, and Steven H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002. doi:10.1073/pnas.012582999.

32 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Fast distributed algorithms for connectivity and mst in large graphs. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA'16, pages 429–438, 2016. doi:10.1145/2935764.2935785.

33 David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed mst construction. In *40th Annual Symposium on Foundations of Computer Science*, FOCS'99, pages 253–261, 1999. doi:10.1109/SFFCS.1999.814597.

34 Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001. doi:10.1007/3-540-45518-3_18.

35 Robert E. Tarjan and Uzi Vishkin. An efficient parallel biconnectivity algorithm. *SIAM J. Comput.*, 14(4):862–874, 1985. doi:10.1137/0214061.

36 Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998. doi:10.1038/30918.

37 Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004. doi:10.1109/JSAC.2003.818784.

Randomized Rumor Spreading Revisited

Benjamin Doerr[1] and Anatolii Kostrygin[2]

1 **Laboratoire d'Informatique (LIX), École Polytechnique, Palaiseau, France**
doerr@lix.polytechnique.fr
2 **Laboratoire d'Informatique (LIX), École Polytechnique, Palaiseau, France**
anatolii.kostrygin@gmail.com

Abstract

We develop a simple and generic method to analyze randomized rumor spreading processes in fully connected networks. In contrast to all previous works, which heavily exploit the precise definition of the process under investigation, we only need to understand the probability and the covariance of the events that uninformed nodes become informed. This universality allows us to easily analyze the classic push, pull, and push-pull protocols both in their pure version and in several variations such as messages failing with constant probability or nodes calling a random number of others each round. Some dynamic models can be analyzed as well, e.g., when the network is a $G(n,p)$ random graph sampled independently each round [Clementi et al. (ESA 2013)].

Despite this generality, our method determines the expected rumor spreading time precisely apart from additive constants, which is more precise than almost all previous works. We also prove tail bounds showing that a deviation from the expectation by more than an additive number of r rounds occurs with probability at most $\exp(-\Omega(r))$.

We further use our method to discuss the common assumption that nodes can answer any number of incoming calls. We observe that the restriction that only one call can be answered leads to a significant increase of the runtime of the push-pull protocol. In particular, the double logarithmic end phase of the process now takes logarithmic time. This also increases the message complexity from the asymptotically optimal $\Theta(n \log\log n)$ [Karp, Shenker, Schindelhauer, Vöcking (FOCS 2000)] to $\Theta(n \log n)$. We propose a simple variation of the push-pull protocol that reverts back to the double logarithmic end phase and thus to the $\Theta(n \log\log n)$ message complexity.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Epidemic algorithm, rumor spreading, dynamic graph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.138

1 Introduction

Randomized rumor spreading is one of the core primitives to disseminate information in distributed networks. It builds on the paradigm that nodes call random neighbors and exchange information with these contacts. This gives highly robust dissemination algorithms belonging to the broader class of gossip-based algorithms that, due to their epidemic nature, are surprisingly efficient and scalable. Randomized rumor spreading has found numerous applications, among others, maintaining the consistency of replicated databases [9], disseminating large amounts of data in a scalable manner [25], and organizing any kind of communication in highly dynamic and unreliable networks like wireless sensor networks and mobile ad-hoc networks [21]. Randomized rumor spreading processes are also

used to model epidemic processes like viruses spreading over the internet [1], news spreading in social networks [10], or opinions forming in social networks [24].

The importance of these processes not only has led to a huge body of experimental results, but, starting with the influential works of Frieze and Grimmett [16] and Karp, Shenker, Schindelhauer, and Vöcking [22] also to a large number of mathematical analyses of rumor spreading algorithms giving runtime or robustness guarantees for existing algorithms and, based on such findings, proposing new algorithms.

Roughly speaking, two types of results can be found in the literature, general bounds trying to give a performance guarantee based only on certain graph parameters and analyses for specific graphs or graph classes. In the domain of general bounds, there is the classic maximum-degree-diameter bound of [13] and more recently, a number of works bounding the rumor spreading time in terms of conductance or other expansion properties [26, 6, 18, 19], which not only greatly helped our understanding of existing processes, but could also be exploited to design new dissemination algorithms [2, 3, 4, 20]. The natural downside of such general results is that they often do not give sharp bounds. It seems that among the known graph parameters, none captures very well how suitable this network structure is for randomized rumor spreading. Also, it has to be mentioned that these results mostly apply to the push-pull protocol.

The other research direction followed in the past is to try to prove sharper bounds for specific graph classes. This led, among others, to the results that the push-protocol spreads a rumor in a complete graph in time $\log_2 n + \ln n \pm \omega(1)$ with high probability $1 - o(1)$ (whp.) [28] (and in time $\log_{2-p} n + \frac{1}{p} \ln n \pm o(\log n)$ when messages fail independently with probability p), whereas the push-pull protocol does so in time $\log_3 n + O(\log \log n)$ [22]. The push protocol spreads rumors in hypercubes in time $O(\log n)$ whp. [13], determining the leading constant is a major open problem. For Erdős-Rény random graphs with edge probability asymptotically larger than the connectivity threshold, again a runtime of $\log_{2-p} n + \frac{1}{p} \ln n \pm o(\log n)$ was shown for the push protocol allowing transmission errors with rate p [14]. For preferential attachment graphs, which are often used as model for real-world networks, it was proven that the push-protocol needs $\Omega(n^\alpha)$ rounds, $\alpha > 0$ some constant, whereas the push-pull protocol takes time $\Theta(\log n)$ and $\Theta((\log n)/\log \log n)$ when nodes avoid to call the same neighbor twice in a row [5, 10]. Even faster rumor spreading times were shown on Chung-Lu power-law random graphs [15].

One weakness of all these results on specific graphs is that they very much rely on the particular properties of the protocol under investigation. Even in fully connected networks (complete graphs), the existing analyses for the basic push protocol [16, 28, 12], the push protocol in the presence of transmission failures [11], the push protocol with multiple calls [27], and the push-pull protocol [22] all uses highly specific arguments that cannot be used immediately for the other processes. This is despite the fact that the global behavior of these processes is often very similar. For example, all processes mentioned have an exponential expansion phase in which the number of informed node roughly grows by a constant factor until a constant fraction of the nodes is informed. Clearly, this hinders a faster development of the field. Note that the typical analysis of a rumor spreading protocol in the papers cited above needs between six and eight pages of proofs.

1.1 Our Results

In this work, we make a big step forward towards overcoming this weakness. We propose a *general analysis method* for all symmetric and memoryless rumor spreading processes in complete networks. It allows to easily analyze all rumor spreading processes mentioned above

and many new ones. The key to this generality is showing that the rumor spreading times for these protocols are determined by the probabilities p_k of a new node becoming informed in a round starting with k informed nodes together with a mild bound on the covariance on the indicator random variables of the events that new nodes become informed. Consequently, all other particularities of the protocol can safely be ignored.

Despite this generality, our method gives bounds for the expected rumor spreading time that are *tight apart from an additive constant number of rounds*. Such tight bounds so far have only been obtained once, namely for the basic push protocol [12].

Our method also gives *tail bounds* stating that deviations from the expectation by an additive number of at least r of rounds occur with probability at most $A' \exp(-\alpha' r)$, where $A', \alpha' > 0$ are absolute constants. Such a precise tail bound was previously given only for the push protocol in [12]. Note that our tail bounds imply the usual whp-statements, e.g., that overshooting the expectation by any $\omega(1)$ term happens with probability $o(1)$ only, and that a rumor spreading time of $O(\log n)$ can be obtained with probability $1 - n^{-c}$, c any constant, by making the implicit constant in the time bound large enough.

We use our method to obtain the following particular results. We only state the expected runtimes. In all cases, the above tail bounds are valid as well.

Classic protocols, robustness: We start by analyzing the three basic push, pull, and push-pull protocols. In the *push protocol*, in each round each informed node calls a random node and sends a copy of the rumor to it. In the *pull protocol*, in each round each uninformed node calls a random node and tries to obtain the rumor from it. In the *push-pull protocol*, all nodes contact a random node and in each such contact the informed nodes send rumor to the communication partner.

For these three protocols, both in the fault-free setting and when assuming that calls fail independently with probability $1 - p$, our method easily yields the expected rumor spreading times given in Table 1. Note that all previous works apart from [12] did not state explicitly a bound for the expected runtime. Note further that for half of the settings regarded in Table 1 no previous result existed. In particular, we are the first to find that the double logarithmic shrinking phase observed by Karp et al. [22] for the push-pull protocol disappears when messages fail with constant probability p, and is instead replaced by an ordinary shrinking regime with the number of uninformed nodes reducing by roughly a factor of $(1-p)e^{-p}$ each round. This observation is not overly deep, but has the important consequence that the message complexity of the push-pull protocol raises from the theoretically optimal $\Theta(n \log \log n)$ value proven in [22] to an order of magnitude of $\Theta(n \log n)$ in the presence of a constant rate of transmission errors. Hence the significant superiority of the push-pull protocol over the push protocol in the fault-free setting reduces to a constant-factor advantage in the faulty setting.

Multiple calls: Panagiotou, Pourmiri, and Sauerwald [27] proposed a variation of the classic protocols in which the number of calls (always to different nodes) each node performs when active is a positive random variable R. They mostly assume that for each node, this random number is sampled once at the beginning of the process. For the case that R has constant expectation and variance, they show that the rumor spreading time of the push protocol is $\log_{1+\mathbb{E}[R]} n + \frac{1}{\mathbb{E}[R]} \ln n \pm o(\log n)$ with high probability and that the rumor spreading time of the push-pull protocol is $\Omega(\log n)$ with probability $1 - \varepsilon$, $\varepsilon > 0$. When R follows a power law with exponent $\beta = 3$, the push-pull protocol takes $\Theta(\frac{\log n}{\log \log n})$ rounds, and when $2 < \beta < 3$, it takes $\Theta(\log \log n)$ rounds.

Table 1 New and previous-best results for rumor spreading time T of the classic rumor spreading protocols in complete graphs on n vertices. The first line of each table entry contains the result that follows from the method proposed in this work, the second line states the best previous result (if any). For all new bounds on the expected rumor spreading time, a tail bound of type $\mathbb{P}[|T \geq \mathbb{E}[T]| \geq r] \leq A' \exp(-\alpha' r)$ with $A', \alpha' > 0$ suitable constants follows as well from this work. In [12], such a bound was given for the rumor spreading time of the push protocol without transmission failures.

	no transmission failures	calls fail indep. with prob. $1-p \in (0,1)$
push protocol	$\mathbb{E}[T] = \log_2 n + \ln n \pm O(1)$ $\lfloor \log_2 n \rfloor + \ln n - 1.116 \leq \mathbb{E}[T] \leq \lceil \log_2 n \rceil + \ln n + 2.765 + o(1)$ [12]	$\mathbb{E}[T] = \log_{1+p} n + \frac{1}{p} \ln n \pm O(1)$ $T = \log_{1+p} n + \frac{1}{p} \ln n \pm o(\log n)$ whp. [11]
pull protocol	$\mathbb{E}[T] = \log_2 n + \log_2 \ln n \pm O(1)$	$\mathbb{E}[T] = \log_{1+p} n + \frac{1}{\ln \frac{1}{1-p}} \ln n \pm O(1)$
push-pull protocol	$\mathbb{E}[T] = \log_3 n + \log_2 \ln n \pm O(1)$ $T = \log_3 n \pm O(\log \log n)$ whp. [22]	$\mathbb{E}[T] = \log_{1+2p} n + \frac{1}{p + \ln \frac{1}{1-p}} \ln n \pm O(1)$

The model of [27] makes sense when assuming that nodes have generally different communication capacities. To model momentarily different capacities, e.g., caused by being occupied with other communication tasks, we assume that the random variable is resampled for each node in each round. We also allow R to take the value 0. Again for the case $\mathbb{E}[R] = \Theta(1)$ and $\mathrm{Var}[R] = O(1)$, we show that the expected rumor spreading time of the push protocol is $\log_{1+\mathbb{E}[R]} n + \frac{1}{\mathbb{E}[R]} \ln n \pm O(1)$. The rumor spreading time of the push-pull protocol depends critically on the smallest value ℓ which R takes with positive probability. If $\ell = 0$, that is, with constant probability nodes contact no other node, then there is no double exponential shrinking and the expected rumor spreading time is $\log_{1+2\mathbb{E}[R]} n + \frac{1}{\mathbb{E}[R] - \ln \mathbb{P}[R=0]} \ln n \pm O(1)$. If nodes surely perform at least one call, then we have a double exponential shrinking regime and an expected rumor spreading time of $\log_{1+2\mathbb{E}[R]} n + \log_{1+\ell} \ln n \pm O(1)$.

Dynamic networks: We also show that our method is capable of analyzing dynamic networks when the dynamic is memory-less. Clementi et al. [7] have shown that when the network in each round is a newly sampled $G(n,p)$ random graph, then for any constant c the rumor spreading time of the push protocol is $\Theta(\log(n)/\min\{p, 1/n\})$ with probability $1 - n^{-c}$. We sharpen this result for the most interesting regime that $p = a/n$, a a positive constant. For this case, we show that the expected rumor spreading time is $\log_{2-e^{-a}} n + \frac{1}{1-e^{-a}} \ln(n) + O(1)$. Our tail bound $\mathbb{P}[|T - \mathbb{E}[T]| \geq r] \leq A' \exp(-\alpha' r)$ for suitable constants $A', \alpha > 0$ implies also the large deviation statement of [7] (where for $\Theta(\log n)$ deviations in the lower tail the trivial $\log_2(n)$ lower bound holding with probability 1 should be used).

Answering single calls only: We finally use our method to discuss an aspect mostly ignored by previous research. While in all protocols above (apart from the one of [27]) it is assumed that each node can call at most one other node per round, it is tacitly assumed in the pull and push-pull protocols that nodes can answer all incoming calls. For complete graphs on n vertices, the classic balls-into-bins theory immediately gives that in a typical round there is at least one node that receives $\Theta(\frac{\log n}{\log \log n})$ calls. So unlike for the outgoing traffic, nodes are implicitly assumed to be able to handle very different amounts of incoming traffic in one round.

The first to discuss this issue are Daum, Kuhn, and Maus [8] (also the SIROCCO 2016 best paper). Among other results, they show that if only one incoming call can be answered and if this choice is taken adversarially, then there are networks where a previously polylogarithmic

rumor spreading time of the pull protocol becomes $\tilde{\Omega}(\sqrt{n})$. If the choice which incoming call is answered is taken randomly, then things improve and the authors show that for any network, the rumor spreading times of the pull and push-pull protocol increase by at most a factor of $O(\frac{\Delta(G)}{\delta(G)} \log n)$ compared to the variant in which all incoming calls are answered. Subsequently, Ghaffari and Newport [17] showed that with the restriction to accept only one incoming call, the general performance guarantees for the push-pull protocol in terms of vertex expansion or conductance [18, 19] do not hold. Kiwi and Caro [23] showed that solving the problem of multiple incoming calls via a FIFO queue can lead to extremely long rumor spreading times.

With our generic method, we can easily analyze this aspect of rumor spreading on complete graphs. While for the pull protocol only the growth phase mildly slows down, giving a total expected rumor spreading time of $\mathbb{E}[T] = \log_{2-1/e} n + \log_2 \ln n \pm O(1)$, for the push-pull protocol also the double logarithmic shrinking phase breaks down and we observe a total runtime of $\mathbb{E}[T] = \log_{3-2/e} n + \frac{1}{2} \ln n \pm O(1)$ and, similarly as for the push-pull protocol with transmission failures, an increase of the message complexity to $\Theta(n \log n)$. The reason, as our proof reveals, is that when a large number of nodes are informed, then their push calls have little positive effect (as in the classic push-pull protocol), but they now also block other nodes' pull calls from being accepted. This problem can be overcome by changing the protocol so that informed nodes stop calling others when the rumor is $\log_{3-2/e} n$ rounds old. The rumor spreading time of this modified push-pull protocol is $\mathbb{E}[T] = \log_{3-2/e} n + \log_2 \ln n \pm O(1)$ and, when halted at the right moment, this process takes $\Theta(n \log \log n)$ messages.

2 Outline of the Analysis Method

As just discussed, the main advantages of our approach are its universality and the very tight bounds it proves. We now briefly sketch the main new ideas that lead to this progress. Interestingly, they are rather simpler than the ones used in previous works.

2.1 Tight Bounds via a Target-Failure Calculus

We first describe how we obtain estimates for the rumor spreading time that are *tight apart from additive constants*. Let us take as example the classic push protocol. It is easy to compute that in a round starting with k informed nodes, the expected number of newly informed nodes is $E(k) = k - \Theta(k^2/n)$. Hence roughly speaking the number of informed nodes doubles each round (which explains the $\log_2 n$ part of the $\log_2 n + \ln n \pm O(1)$ rumor spreading time), but there is a growing gap to truly doubling caused by (i) calls reaching already informed nodes and (ii) several calls reaching the same target. This weakening of the doubling process was a main difficulty in all previous works.

The usual way to analyze this weakening doubling process is to partition the rumor spreading process in phases and within each phase to uniformly estimate the progress. For example, Pittel [28] considers 7 phases. He argues first that with high probability the number if informed nodes doubles until $n_1 = o(\sqrt{n})$ nodes are informed. Then, until $n_2 = n/\log^2(n)$ nodes are informed, with high probability in each round the number of informed nodes increases by at least a factor of $2(1 - \frac{1}{\log^2(n)})$. Consequently, this second phase lasts at most $\log_{2(1-\frac{1}{\log^2(n)})}(n_2/n_1)$ rounds. While this type of argument gives good bounds for phases bounded away from the middle regime with both $\Theta(n)$ nodes informed and uninformed, we do not see how this "estimating a phase uniformly" argument can cross the middle regime without losing a number $\omega(1)$ of rounds.

For this reason, we proceed differently. To prove upper bounds on rumor spreading times, for each number k of informed nodes, we formulate a pessimistic round target $E_0(k)$ that is sufficiently below the expected number $E(k)$ of newly informed nodes. Here "sufficiently below" means that the probability $q(k)$ to fail reaching this target number of informed nodes is small, but not necessarily $o(1)$ as in all previous analyses. Using a restart argument, we observe that the random time needed to go from k informed nodes to at least $E_0(k)$ informed nodes is stochastically dominated by 1 plus a geometric random variable with parameter $1 - q(k)$, where all our geometric random variables count the number of failures until success (this is one of the two definitions of geometric distributions that are in use). In particular, the expected time to go from k to at least $E_0(k)$ informed nodes is at most $1 + \frac{q(k)}{1-q(k)}$.

The second, again elementary, key argument is that when we define a sequence of round targets by $k_0 := 1$, $k_1 := E_0(k_0)$, $k_2 := E_0(k_1), \dots$ with suitably defined $E_0(\cdot)$, then the k_i grow almost like 2^i (in the example of the classic push protocol). More precisely, there is a $T = \log_2 n \pm O(1)$ such that $k_T = \Theta(n)$. Hence together with the previous paragraph we obtain that the number of rounds to reach k_T informed nodes is dominated by T plus a sum of independent geometric random variables. This sum has expectation $\sum_{i=0}^{T-1} \frac{q(k_i)}{1-q(k_i)} = O(\sum_{i=0}^{T-1} q(k_i))$, so it suffices that the sum of the failure probabilities $q(k_i)$ is a constant (unlike in previous works, where it needed to be $o(1)$). A closer look at this sum also gives the desired tail bounds.

Similarly, to prove matching lower bounds, we define optimistic round targets $E_0(k)$ such that a round starting with k informed nodes finds it unlikely to reach $E_0(k)$ informed nodes. Since again we want to allow failure probabilities that are constant, we now have to be more careful and also quantify the probability to overshoot $E_0(k)$ by larger quantities. This will then allow to argue that when defining a sequence of round targets recursively as above, then the expected number of targets overjumped (and thus the expected number of rounds saved compared to the "one target per round" calculus), is only constant.

We remark that a target-failure argument similar to ours was used already in [12], there however only to give an upper bound for the runtime of the push protocol in the regime from n^s, s a small constant, to $\Theta(n)$ informed nodes, that is, the later part of the exponential growth regime of the push process, in which via Chernoff bounds very strong concentration results could be exploited. Hence the novelty of this work with respect to the target-failure argument is that this analysis method can be used (i) also from the very beginning of the process on, where we have no strong concentration, (ii) also for the exponential and double exponential shrinking regimes of rumor spreading processes, and (iii) also for lower bounds.

2.2 Uniform Treatment of Many Rumor Spreading Processes

As discussed earlier, the previous works regarding different rumor spreading processes on complete graphs all had to use different arguments. The reason is that the processes, even when looking similar from the outside, are intrinsically different when looking at the details. As an example, let us consider the first few rounds of the push and the pull protocol. In the push protocol, we just saw that while there are at most $o(\sqrt{n})$ nodes informed, then a birthday paradox type argument gives that with high probability we have perfect doubling in each round. For the pull process, in which each uninformed node calls a random node and becomes informed when the latter was informed, we also easily compute that a round starting with k informed nodes creates an expected number of $(n-k)\frac{k}{n} = k - \frac{k^2}{n}$ newly informed nodes. However, since these are binomially distributed, there is no hope for perfect doubling. In fact, for the first constant number of rounds, we even have a constant probability that not a single node becomes informed.

The only way to uniformly treat such different processes is by making the analysis depend only on general parameters of the process as opposed to the precise definition. Our second main contribution is distilling a few simple conditions that (i) subsume essentially all symmetric and time-invariant rumor spreading processes on complete graphs and (ii) suffice to prove rumor spreading times via the above described target-failure method. All this is made possible by the observation that the target-failure method needs much less in terms of failure probabilities than previous approaches, in particular, it can tolerate constant failure probabilities. Consequently, instead of using Chernoff and Azuma bounds for independent or negatively correlated random variables (which rely on the precise definition of the process), it suffices to use Chebyshev's inequality as concentration result.

Consequently, to apply our method we only need to (i) understand (with a certain precision) the probability p_k that an uninformed node becomes informed in a round starting with k informed nodes; recall that we assumed symmetry, that is, this probability is the same for all uninformed nodes, and (ii) we need to have a mild upper bound on the covariance of the indicator random variables of the events that two nodes become informed.

The probabilities p_k usually are easy to compute from the protocol definition. Also, we do not know them precisely. For example, for the growth phase of the push protocol discussed above, it suffices to know that there are constants $a < 2$ and a' such that for all $k < n/2$ we have $\frac{k}{n}(1 - a\frac{k}{n}) \leq p_k \leq \frac{k}{n}(1 + a'\frac{k}{n})$. This (together with the covariance condition) is enough to show that the rumor spreading process takes $\log_2 n \pm O(1)$ rounds to inform $n/2$ nodes or more. The constants a, a' have no influence on the final result apart from the additive constant number of rounds hidden in the $O(1)$ term. The covariances are also often easy to bound with sufficient precision, among others, because many in processes the events that two uniformed nodes become informed are independent or negatively correlated.

In our general analysis method, we profit from the fact that seemingly all reasonable rumor spreading processes in complete networks can be described via three regimes:

Exponential growth: Up to a constant fraction fn of informed nodes, $p_k = \gamma_n \frac{k}{n}(1 \pm O(\frac{k}{n}))$. The number of informed nodes thus increases roughly by a factor of $(1 + \gamma_n)$ in each round, hence the expected time to reach fn informed nodes or more is $\log_{1+\gamma_n} n \pm O(1)$.

Exponential shrinking: From a certain constant fraction $u = n - k = gn$ of uninformed nodes on, the probability of remaining uninformed satisfies $1 - p_{n-k} = e^{-\rho_n} \pm O(\frac{u}{n})$. This leads to a shrinking of the number of uninformed nodes by essentially a factor of $e^{-\rho_n}$ per round. Hence when starting with gn informed nodes, it takes another $\frac{1}{\rho_n} \ln n \pm O(1)$ rounds in expectation until all are informed.

Double exponential shrinking: From a certain constant fraction $u = n - k = gn$ of uninformed nodes on, the probability of remaining uninformed satisfies $1 - p_{n-k} = \Theta((\frac{u}{n})^{\ell-1})$. Now the expected time to go from gn uninformed nodes to no uninformed node is $\log_\ell \ln n \pm O(1)$.

Due to their different nature, we cannot help treating these three regimes separately, however all with the target-failure method. Hence the main differences between these regimes lie in defining the pessimistic estimates for the targets, computing the failure probabilities, and computing the number of intermediate targets until the goal is reached. All this only needs computing expectations, using Chebyshev's inequality, and a couple of elementary estimates.

3 Precise Statement of the Technical Results

In this work, we consider only *homogeneous rumor spreading processes* characterized as follows. We always assume that we have n nodes. Each node can be either *informed* or *uninformed*. We assume that the process starts with exactly one node being informed. Uninformed nodes may become informed, but an informed node never becomes uninformed. We consider a discrete time process, so the process can be partitioned into *rounds*. In each round each uninformed node can become informed. Whenever a round starts with k nodes being informed, then the probability for each uninformed node to become informed is some number p_k, which only depends on the number k of informed nodes at the beginning of the round.

The main insight of this work is that for such homogeneous rumor spreading processes we can mostly ignore the particular structure of the process and only work with the *success probabilities* p_k defined above and the *covariance numbers* c_k defined as follows.

▶ **Definition 1** (Covariance numbers)**.** For a given homogeneous rumor spreading process and $k \in [1..n-1]$ let c_k be the smallest number such that whenever a round starts with k informed nodes and for any two uninformed nodes x_1, x_2, the indicator random variables X_1, X_2 for the events that these nodes become informed in this round satisfy

$$\mathrm{Cov}[X_1, X_2] \leq c_k.$$

Upper bound for these covariances imply upper bounds on the variance of the number of nodes newly informed in a round. If the latter is small, Chebyshev's inequality yields that the actual number of newly informed nodes deviates not a lot from its expectation (which is determined by p_k).

Our main interest is studying after how many round all nodes are informed.

▶ **Definition 2** (Rumor spreading times)**.** Consider a homogeneous rumor spreading process. For all $t = 0, 1, \dots$ denote by I_t the number of informed nodes at the end of the t-th round ($I_0 := 1$). Let $k \leq m \leq n$. By $T(k, m)$ we denote the time it takes to increase the number of informed nodes from k to m or more, that is,

$$T(k, m) = \min\{t - s | I_s = k \text{ and } I_t \geq m\}.$$

We call $T(1, n)$ the rumor spreading time of the process.

As it turns out, almost all homogeneous rumor spreading processes can be analyzed via three regimes.

3.1 Exponential Growth Regime

When not too many nodes are informed, in most rumor spreading processes we observe roughly a constant-factor increase of the number of informed nodes in one round, however, this increase becomes weaker with increasing number of informed nodes.

▶ **Definition 3** (Exponential growth conditions)**.** Let γ_n be bounded between two positive constants. Let $a, b, c \geq 0$ and $0 < f < 1$. We say that a homogeneous rumor spreading process satisfies the *upper (respectively lower) exponential growth conditions* in $[1, fn[$ if for any $n \in \mathbb{N}$ big enough the following properties are satisfied for any $k < fn$.

(i) $p_k \geq \gamma_n \frac{k}{n} \cdot \left(1 - a\frac{k}{n} - \frac{b}{\ln n}\right)$ (respectively $p_k \leq \gamma_n \frac{k}{n} \cdot \left(1 + a\frac{k}{n} + \frac{b}{\ln n}\right)$).

(ii) $c_k \leq c\frac{k}{n^2}$.

In the case of the upper exponential growth condition, we also require $af < 1$.

These growth conditions suffice to prove that in an expected time of at most (respectively at least) $\log_{1+\gamma_n} n \pm O(1)$ rounds a linear number of nodes becomes informed. Consequently, the decrease of the dissemination speed when more nodes are informed (quantified by the term $-a\frac{k}{n}$ in the upper exponential growth condition), which was a main difficulty in previous analyses, has only an $O(1)$ influence on the rumor spreading time.

▶ **Theorem 4.** *If a homogeneous rumor spreading process satisfies the upper (lower) exponential growth conditions in* $[1, fn[$*, then there are constants* $A', \alpha' > 0$ *such that*

$$\mathbb{E}[T(1, fn)] \underset{(\geq)}{\leq} \log_{1+\gamma_n} n \underset{(-)}{+} O(1),$$

$$\mathbb{P}[T(1, fn) \underset{(\leq)}{\geq} \log_{1+\gamma_n} n \underset{(-)}{+} r] \leq A' \exp(-\alpha' r) \text{ for all } r \in \mathbb{N}.$$

When the lower exponential growth conditions are satisfied, then also there is an $f' \in]f, 1[$ *such that with probability* $1 - O\left(\frac{1}{n}\right)$ *at most* $f'n$ *nodes are informed at the end of round* $T(1, fn)$*.*

We note that the upper tail bound is tight apart from the implicit constants. This is witnessed, for example, by the pull protocol, where rounds starting with only a constant number of informed nodes have a constant probability of not informing any new node.

3.2 Exponential Shrinking Regime

In a sense dual to the previous regime, in many rumor spreading processes we observe that the number of uninformed nodes shrinks by a constant factor once sufficiently many nodes are informed. Again, the weaker shrinking at the beginning of this regime has only an $O(1)$ influence on the resulting rumor spreading times.

▶ **Definition 5** (Exponential shrinking conditions)**.** Let ρ_n be bounded between two positive constants. Let $0 < g < 1$, and $a, c \in \mathbb{R}_{\geq 0}$. We say that a homogeneous rumor spreading process satisfies the *upper (respectively lower) exponential shrinking conditions* if for any $n \in \mathbb{N}$ big enough, the following properties are satisfied for all $u = n - k \leq gn$.

(i) $1 - p_k = 1 - p_{n-u} \leq e^{-\rho_n} + a\frac{u}{n}$ (respectively $1 - p_k = 1 - p_{n-u} \geq e^{-\rho_n} - a\frac{u}{n}$).

(ii) $c_k = c_{n-u} \leq \frac{c}{u}$.

For the upper exponential shrinking conditions, we also assume that $e^{-\rho_n} + ag < 1$.

▶ **Theorem 6.** *If a homogeneous rumor spreading process satisfies the upper (lower) exponential shrinking conditions, then there are* $A'\alpha' > 0$ *such that*

$$\mathbb{E}[T(n - \lfloor gn \rfloor, n)] \underset{(\geq)}{\leq} \frac{1}{\rho_n} \ln n \underset{(-)}{+} O(1),$$

$$\mathbb{P}[T(n - \lfloor gn \rfloor, n) \underset{(\leq)}{\geq} \frac{1}{\rho_n} \ln n \underset{(-)}{+} r] \leq A' \exp(-\alpha' r) \text{ for all } r \in \mathbb{N}.$$

Again, the upper tail bound is tight apart from the constants as shown by the last rounds of the push protocol.

3.3 Double Exponential Shrinking Regime

Protocols using pull operations in the absence of transmission failures display a faster reduction of the number if uninformed nodes.

▶ **Definition 7** (Double exponential shrinking conditions). Let $g \in]0,1]$, $\ell > 1$, and $a, c \in \mathbb{R}_{\geq 0}$ such that $ag^{\ell-1} < 1$. We say that a homogeneous rumor spreading process satisfies the *upper (respectively lower) double exponential shrinking conditions* if for any n big enough the following properties are satisfied for all $u = n - k \in [1, gn]$.

(i) $1 - p_{n-u} \leq a\left(\frac{u}{n}\right)^{\ell-1}$ (respectively $1 - p_{n-u} \geq a\left(\frac{u}{n}\right)^{\ell-1}$).

(ii) $c_{n-u} \leq c\frac{n}{u^2}$.

▶ **Theorem 8.** *If a homogeneous rumor spreading process satisfies the upper (lower) double exponential shrinking conditions, then there are $A', \alpha' > 0$ and R (depending on α) such that*

$$\mathbb{E}[T(n - \lfloor gn \rfloor, n)] \underset{(\geq)}{\leq} \log_\ell \ln n \underset{(-)}{+} O(1),$$

$$\mathbb{P}[T(n - \lfloor gn \rfloor, n) \geq \log_\ell \ln n + r] \leq O(n^{-\alpha' r + A'}) \text{ for all } r \in \mathbb{N},$$

$$(\mathbb{P}[T(n - \lfloor gn \rfloor, n) \leq \log_\ell \ln n - R] \leq O(n^{-1+2\ell\alpha})).$$

The last rounds of the push-pull protocol show that the upper tail bound is tight apart from the constants. The lower tail bound is clearly not best possible, but most likely good enough for most purposes.

3.4 Connecting Regimes

While often these above described three regimes suffice to fully analyze a rumor spreading process, occasionally it is necessary or convenient to separately regard a constant number of rounds between the growth and the shrinking regime. This is achieved by the following two lemmas.

▶ **Lemma 9.** *Consider a homogeneous rumor spreading process. Let $0 < \ell < m < n$ and $0 < p < 1$. Suppose for any number $\ell \leq k < m$, we have $p_k \geq p$. Then*

$$\mathbb{E}[T(\ell, m)] \leq \tfrac{n-\ell}{n-m} \cdot \tfrac{1}{p},$$

$$\mathbb{P}[T(\ell, m) > r] \leq \tfrac{n-\ell}{n-m} \cdot (1-p)^r \text{ for all } r \in \mathbb{N}.$$

▶ **Lemma 10.** *Let $f, p \in]0,1[$ and $c > 0$. Suppose that for any $k < fn$ we have $p_k \leq p$ and $c_k \leq \frac{c}{n}$. Then there exists $f' \in]f, 1[$ such that with probability $1 - O\left(\frac{1}{n}\right)$ at the end of some round the number of informed nodes will be between fn and $f'n$.*

4 Applying the Above Technical Results

In this section, we sketch how to use the above tools to obtain some of the results described in Section 1.1. To ease the notation we always assume that nodes call random nodes, that is, including themselves. The main observation is that computing the p_k is usually very elementary. For the covariance conditions, often we easily observe a negative or zero covariance, but when this is not true, then things can become technical.

For the *basic push, pull, and push-pull protocols*, we easily observe that all covariances to be regarded are negative or zero: Knowing that one uninformed node x_1 becomes informed in the current round has no influence on the pull call of another uninformed node x_2. When the protocol has push calls and x_1 was informed via a push call, then this event makes it slightly less likely that x_2 becomes informed via a push call, simply because at least one informed node is occupied with calling x_1.

The success probabilities p_k are easy to compute right from the protocol definition. When k nodes are informed, then the probabilities that an uninformed node becomes informed are

$$p_k = \begin{cases} 1-(1-1/n)^k & \text{for the push protocol,} \\ k/n & \text{for the pull protocol,} \\ p_k = 1-(1-1/n)^k \frac{n-k}{n} & \text{for the push-pull protocol.} \end{cases}$$

Using elementary estimates like $1-k/n \leq (1-1/n)^k \leq 1-k/n+k^2/2n^2$, we see that the push and pull protocols satisfy the exponential growth conditions with $\gamma_n = 1$, whereas the push-pull protocol does the same with $\gamma_n = 2$. The push protocol satisfies the exponential shrinking conditions with $\rho_n = 1$. The pull and push-pull protocols satisfy the double exponential shrinking conditions with $\ell = 2$. All growth conditions are satisfied at least up to $k = n/2$ informed nodes and all shrinking conditions are satisfied at least for $u \leq n/2$ uninformed nodes, so we do not need the intermediate lemmas. This proves our results given in Table 1 for the fault-free case.

Faulty communication: The same arguments (with different constants γ_n and ρ_n) suffice to analyze these protocols when messages get lost independently with probability $1-p$. The only structural difference is that now for the pull and push-pull protocols uninformed nodes remain uninformed with at least constant probability. For this reason, now all three protocols have an exponential shrinking phase.

The push-pull protocol with the restriction that nodes *answer only a single incoming call* randomly chosen among the incoming calls is an example where the exponential growth and shrinking conditions are harder to prove. To compute the p_k we assume that all n calls have a random unique priority in $[1..n]$ and that the call with lowest priority number is accepted. For fixed priority, the probability of being accepted is easy to compute, and this leads to the success probability of a pull call. For the probability to become informed via a push call, the simple argument that the first incoming call is from an informed node with probability k/n solves the problem. When showing the covariance conditions, we face the problem that it is indeed not clear if we have negative or zero covariance. The event that some node becomes informed increases the chance that this node received a push call. This push call cannot interfere with another node's pull call to an informed node. So it does have some positive influence on the probability of another uninformed node to become informed. Fortunately, for our covariance conditions allow some positive correlation. Because of this, very generally speaking, we can ignore certain difficulties to handle situations when they occur rare enough.

Dynamic communication graphs: Being maybe the result where it is most surprising that bounds sharp apart from additive constants can be obtained, we now regard in more detail a problem regarded in [7]. There, the performance of push rumor spreading in a group of n agents was investigated when the actual communication network is changing in each round. As one such dynamic models, it was assumed that the communication graph in each round is a newly sampled $G(n,p)$ random graph, that is, there is an edge independently with probability p between any two vertices. For the ease of presentation, we assume that the edge probability equals $p = a/n$ for some constant $a > 0$. This is clearly the most interesting case. For such (and larger) p, a rumor spreading time of $\Theta(\log n)$ was shown to hold with inverse-polynomial failure probability. Recalling that for $p = a/n$ the graph $G(n,p)$ is not connected and has vertex degrees ranging from 0 to $\Theta(\log(n)/\log\log(n))$, this result is not obvious. Also, observe that the actions of all nodes in one round take place in the same random graph, so there are dependencies that have to be taken into account.

▶ **Theorem 11.** *Let T be the time the push protocol needs to inform n nodes when in each round a newly sampled $G(n,p)$, $p = a/n$, random graph represents the communication network. Then*

$$\mathbb{E}[T] = \log_{2-e^{-a}} n + \tfrac{1}{1-e^{-a}} \ln(n) \pm O(1)$$

and there are constants $A', \alpha' > 0$ such that $\mathbb{P}[|T - \mathbb{E}[T]| \geq r] \leq A' \exp(\alpha' r)$ holds for all $r \in \mathbb{N}$.

Recall that a vertex is isolated with probability $e^{-a} + O(1/n)$. Clearly, an informed vertex when isolated necessarily fails to inform another vertex in this round. The rumor spreading time proven above is the same as the one for the case that the communication network is always a complete graph, but calls fail independently with probability e^{-a}. Hence in a sense the changing topology (with low vertex degrees) is not harmful apart from the effect that it creates isolated vertices with constant rate. We did not expect this.

To prove Theorem 11, we first observe that the covariance properties are fulfilled. By symmetry, we can assume that in a round starting with k informed nodes, we first sample the random graph and decide for each node which neighbor it potentially calls in this round, and only then decide randomly which k nodes are informed and have these call the random neighbor determined before. Conditioning on the outcome of random graph, neighbor choice, and on that nodes x and y are not informed, in the remaining random experiment the events "x becomes informed" and "y becomes informed" clearly are negatively correlated.

Estimating the probability p_k for an uninformed node to become informed in a round starting with k informed nodes, is slightly technical. Since it is unlikely that two neighbors of an uninformed node x are connected by an edge, the main contribution to p_k stems from the case that the informed neighbors of x form an independent set. Conditioning on this outcome of the edges in $\{x\} \cup N(x)$, each informed neighbor of x has an independent probability of roughly $(1-e^{-a})/a$ of calling x, giving (again taking care of the dependencies) a probability of roughly $1 - p_k \approx (1 - \frac{a}{n}\frac{1-e^{-a}}{a})^k$ for the event that no informed node calls x. From this, we estimate $\frac{k}{n}(1-e^{-a})(1 - \frac{k+O(1)}{2n}(1-e^{-a})) \leq p_k \leq \frac{k}{n}(1 - e^{-a} + O(1/n))$, showing that the exponential growth conditions are satisfied with $\gamma_n = 1 - e^{-a}$. Similar arguments, again taking some care for the dependencies that the random graph imposes on the actions of informed neighbors, show that the upper exponential shrinking conditions are satisfied for $\rho_n = 1 - e^{-a}$, whereas the lower exponential shrinking conditions are satisfied with $\rho_n = 1 - e^{-a} + O(\log(n)^2/n)$.

5 Summary, Outlook

In this work, we presented a general, easy-to-use method to analyze homogeneous rumor spreading processes on complete networks (including memoryless dynamic settings). Such processes are important in many applications, among others, due to the use of random peer sampling services in many distributed systems. Such processes also correspond to the fully mixed population model in mathematical epidemiology.

The two main strengths of our method are (i) that it builds only on estimates for the probability and the covariance of the events that new nodes become informed—consequently, many processes can be analyzed with identical arguments (as opposed to all previous works), and (ii) that it determines the expected rumor spreading time precise apart from additive constants (with tail bounds giving in most cases that deviations by an additive number r of rounds occur with probability $\exp(-\Omega(r))$ only). The key to our results is distilling

growth and shrinking conditions which cover essentially all previously regarded homogeneous processes and which imply the desired runtime bounds.

From a broader perspective, this work shows that the traditional approach to randomized processes of splitting the analysis in several phases and then trying to understand each phase with uniform arguments might not be the ideal way to capture the nature of processes with a behavior changing continuously over time. While we demonstrated that the more careful round-target approach is better suited for homogeneous rumor spreading processes, one can speculate if similar ideas are profitable for other randomized algorithms or processes regarded in computer science.

References

1 Noam Berger, Christian Borgs, Jennifer T. Chayes, and Amin Saberi. On the spread of viruses on the internet. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 301–310. SIAM, 2005. doi:10.1145/1070432.1070475.

2 Keren Censor-Hillel, Bernhard Haeupler, Jonathan A. Kelner, and Petar Maymounkov. Global computation in a poorly connected world: fast rumor spreading with no dependence on conductance. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC)*, pages 961–970, 2012. doi:10.1145/2213977.2214064.

3 Keren Censor-Hillel and Hadas Shachnai. Partial information spreading with application to distributed maximum coverage. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 161–170, 2010. doi:10.1145/1835698.1835739.

4 Keren Censor-Hillel and Hadas Shachnai. Fast information spreading in graphs with large weak conductance. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 440–448, 2011. doi:10.1137/1.9781611973082.35.

5 Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumor spreading in social networks. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 375–386. Springer, 2009. doi:10.1007/978-3-642-02930-1_31.

6 Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 399–408. ACM, 2010. doi:10.1145/1806689.1806745.

7 Andrea E. F. Clementi, Pierluigi Crescenzi, Carola Doerr, Pierre Fraigniaud, Francesco Pasquale, and Riccardo Silvestri. Rumor spreading in random evolving graphs. *Random Structures and Algorithms*, 48:290–312, 2016. doi:10.1002/rsa.20586.

8 Sebastian Daum, Fabian Kuhn, and Yannic Maus. Rumor spreading with bounded indegree. *CoRR*, abs/1506.00828, 2015.

9 Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12. ACM, 1987. doi:10.1145/41840.41841.

10 Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Social networks spread rumors in sublogarithmic time. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 21–30. ACM, 2011. doi:10.1145/1993636.1993640.

11 Benjamin Doerr, Anna Huber, and Ariel Levavi. Strong robustness of randomized rumor spreading protocols. *Discrete Applied Mathematics*, 161:778–793, 2013. doi:10.1016/j.dam.2012.10.014.

12 Benjamin Doerr and Marvin Künnemann. Tight analysis of randomized rumor spreading in complete graphs. In *Proceedings of the Eleventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 82–91. SIAM, 2014. doi:10.1137/1.9781611973204.8.

13 Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Randomized broadcast in networks. *Random Structures and Algorithms*, 1:447–460, 1990. doi:10.1002/rsa.3240010406.

14 Nikolaos Fountoulakis, Anna Huber, and Konstantinos Panagiotou. Reliable broadcasting in random networks and the effect of density. In *Proceedings of the 29th International Conference on Computer Communications (INFOCOM)*, pages 2552–2560. IEEE, 2010. doi:10.1109/INFCOM.2010.5462084.

15 Nikolaos Fountoulakis, Konstantinos Panagiotou, and Thomas Sauerwald. Ultra-fast rumor spreading in social networks. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1642–1660. SIAM, 2012.

16 Alan M. Frieze and Geoffrey R. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10:57–77, 1985. doi:10.1016/0166-218X(85)90059-9.

17 Mohsen Ghaffari and Calvin Newport. How to discreetly spread a rumor in a crowd. *CoRR*, abs/1607.05697, 2016.

18 George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 57–68. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2011. doi:10.4230/LIPIcs.STACS.2011.57.

19 George Giakkoupis. Tight bounds for rumor spreading with vertex expansion. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 801–815. SIAM, 2014. doi:10.1137/1.9781611973402.59.

20 Bernhard Haeupler. Simple, fast and deterministic gossip and rumor spreading. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 705–716, 2013. doi:10.1137/1.9781611973105.51.

21 Konrad Iwanicki and Maarten van Steen. Gossip-based self-management of a recursive area hierarchy for large wireless sensornets. *IEEE Transactions on Parallel and Distributed Systems*, 21:562–576, 2010. doi:10.1109/TPDS.2009.89.

22 Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 565–574. IEEE, 2000.

23 Marcos A. Kiwi and Christopher Thraves Caro. FIFO queues are bad for rumor spreading. *IEEE Trans. Information Theory*, 63:1159–1166, 2017. doi:10.1109/TIT.2016.2632153.

24 Jon M. Kleinberg. The convergence of social and technological networks. *Communications of the ACM*, 51:66–72, 2008. doi:10.1145/1400214.1400232.

25 Miguel Matos, Valerio Schiavoni, Pascal Felber, Rui Oliveira, and Etienne Riviere. BRISA: combining efficiency and reliability in epidemic data dissemination. In *26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 983–994, 2012. doi:10.1109/IPDPS.2012.92.

26 Damon Mosk-Aoyama and Devavrat Shah. Fast distributed algorithms for computing separable functions. *IEEE Trans. Information Theory*, 54:2997–3007, 2008. doi:10.1109/TIT.2008.924648.

27 Konstantinos Panagiotou, Ali Pourmiri, and Thomas Sauerwald. Faster rumor spreading with multiple calls. *The Electronic Journal of Combinatorics*, 22:P1.23, 2015.

28 Boris Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47:213–223, 1987.

Randomized Load Balancing on Networks with Stochastic Inputs*†

Leran Cai[1] and Thomas Sauerwald[2]

1 **University of Cambridge, Cambridge, UK**
`lc647@cl.cam.ac.uk`
2 **University of Cambridge, Cambridge, UK**
`tms41@cl.cam.ac.uk`

Abstract

Iterative load balancing algorithms for indivisible tokens have been studied intensively in the past, e.g., [21, 18, 24]. Complementing previous worst-case analyses, we study an average-case scenario where the load inputs are drawn from a fixed probability distribution. For cycles, tori, hypercubes and expanders, we obtain almost matching upper and lower bounds on the discrepancy, the difference between the maximum and the minimum load. Our bounds hold for a variety of probability distributions including the uniform and binomial distribution but also distributions with unbounded range such as the Poisson and geometric distribution. For graphs with slow convergence like cycles and tori, our results demonstrate a substantial difference between the convergence in the worst- and average-case. An important ingredient in our analysis is a new upper bound on the t-step transition probability of a general Markov chain, which is derived by invoking the evolving set process.

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases random walks, randomized algorithms, parallel computing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.139

1 Introduction

In the last decade, large parallel networks became widely available for industrial and academic users. An important prerequisite for their efficient usage is to balance their work efficiently. Load balancing is known to have applications to scheduling [27], routing [9], numerical computation such as solving partial differential equations [29, 28, 26], and finite element computations [13]. In the standard abstract formulation of load balancing, processors are represented by nodes of a graph, while links are represented by edges. The objective is to balance the load by allowing nodes to exchange loads with their neighbors via the incident edges. In this work we will study a decentralized and iterative load balancing protocol where a processor knows only its current load and that of the neighboring processors. Based on this, decides how much load should be sent (or received).

Load Balancing Models. A widely used approach is diffusion, e.g., the first-order-diffusion scheme [9, 18], where the amount of load sent along each edge in each round is proportional to the load difference between the incident nodes. In this work, we consider the alternative, the so-called matching model, where in each round only the edges of a matching are used

* See [8] for a full version of this work, `https://arxiv.org/abs/1703.08702`.
† The second author was supported by the ERC Starting Grant "Dynamic March".

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 139; pp. 139:1–139:14


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

to average the load locally. In comparison to diffusion, the matching model reduces the communication in the network and moreover tends to behave in a more "monotone" fashion than diffusion, since it avoids concurrent load exchanges which may increase the maximum load or decrease the minimum load in certain cases.

We measure the smoothness of the load distribution by the so-called *discrepancy*, which is the difference between the maximum and minimum load among all nodes. In view of more complex scenarios where jobs are eventually removed or new jobs are generated, the discrepancy seems to be a more appropriate measure than the *makespan*, which only considers the maximum load.

Many studies in load balancing assume that load is arbitrarily divisible. In this so-called *continuous case*, load balancing corresponds to a Markov chain on the graph and one can resort to a wide range of established techniques to analyze the convergence speed [6, 11, 18]. In particular, the *spectral gap* captures the time to reach a small discrepancy fairly accurately, e.g., see [25, 21] for the diffusion and see [7, 17] for the matching model.

However, in many applications a processor's load may consist of tasks which are not further divisible. That is why the continuous case has been also referred to as "idealized case" [21]. A natural way to model indivisible tasks is the *unit-size token model* where one assumes a smallest load entity, the unit-size token, and load is always represented by a multiple of this smallest entity. In the following, we will refer to the unit-size token model as the *discrete case*.

Initiated by the work of [21], there has been a number of studies on load balancing in the discrete case. Unlike the deterministic rounding in [21], [24] analyzed a randomized rounding based strategy, meaning that an excess token will be distributed uniformly at random among the two communicating nodes. The authors of [24] proved that with this strategy the time to reach constant discrepancy in the discrete case is essentially the same as the corresponding time in the continuous case. Their results hold both for the *random matching model*, where in each round a new random matching is generated by a simple distributed protocol, and the *balancing circuit model* (a.k.a. dimension exchange), where a fixed sequence of matching is applied periodically. In this work, we will focus on the *balancing circuit model*, which is particularly well suited for highly structured graphs such as cycles, tori or hypercubes.

Worst-Case vs. Average-Case Inputs. Previous work has almost always adopted the usual worst-case framework for deriving bounds on the load discrepancy [21]. That means any upper bound on the discrepancy holds for an arbitrary input, i.e., an arbitrary initial load vector. While it is of course very natural and desirable to have such general bounds, the downside is that for graphs with poor expansion like cycles or 2D-tori, the convergence is rather slow, i.e., quadratic or linear in the number of nodes n. This serves as a *motivation* to explore an average-case input. Specifically, we assume that the number of load items at each node is sampled independently from a fixed distribution. Our main results demonstrate that the convergence of the load vector is considerably quicker (measured by the load discrepancy), especially on networks with slow convergence in the worst-case such as cycles and 2D-tori.

We point out that many related problems including scheduling on parallel machines or load balancing in a dynamic setting (meaning that jobs are continuously added and processed) have been studied under random inputs, e.g., [3, 12, 2]. To the best of our knowledge, only very few works have studied this question in iterative load balancing. One exception is [22], which investigated the performance of continuous load balancing on tori in the diffusion model. In contrast to this work, however, only upper bounds are given and they hold for the multiplicative ratio between maximum and minimum load, rather than the discrepancy.

Another related work is [4], which presents a distributed algorithm for community detection that is based on averaging a random $\{-1, 1\}$ initial load vector.

1.1 Notation and Background

We assume that $G = (V, E)$ is an undirected, connected graph with n nodes labeled in $[0, n-1]$. Unless stated otherwise, all logarithms are to the base e. The notations $\mathbb{P}[\mathcal{E}]$ and $\mathbb{E}[X]$ denote the probability of an event $\mathcal{E}$ and the expectation of a random variable X, respectively. For any n-dimensional vector x, $\mathrm{disc}(x) = \max_i x_i - \min_i x_i$ denotes the *discrepancy*. By $\frac{\mathbf{1}}{\mathbf{n}}$ we denote the vector with all values being $\frac{1}{n}$.

Matching Model. In the *matching model* (sometimes also called *dimension exchange model*), every two matched nodes in round t balance their load as evenly as possible. This can be expressed by a symmetric n by n matching matrix $\mathbf{M}^{(t)}$, where with slight abuse of notation we use the same symbol for the matching and the corresponding matching matrix. Formally, matrix $\mathbf{M}^{(t)}$ is defined by $\mathbf{M}^{(t)}_{u,u} = 1/2$, $\mathbf{M}^{(t)}_{v,v} = 1/2$ and $\mathbf{M}^{(t)}_{u,v} = \mathbf{M}^{(t)}_{v,u} = 1/2$ if $\{u, v\} \in \mathbf{M}^{(t)} \subseteq E$, and $\mathbf{M}^{(t)}_{u,u} = \mathbf{M}^{(t)}_{v,v} = 1$, $\mathbf{M}^{(t)}_{u,v} = 0$ $(u \neq v)$ if u, v are not matched.

Balancing Circuit. In the *balancing circuit model*, a specific sequence of matchings is applied periodically. More precisely, let $\mathbf{M}^{(1)}, \ldots, \mathbf{M}^{(d)}$ be a sequence of d matching matrices, also called *period* [1]. Then in step $t \geq 1$, we apply the matching matrix $\mathbf{M}^{(t)} := \mathbf{M}^{(((t-1) \mod d)+1)}$. We define the *round matrix* by $\mathbf{M} := \prod_{s=1}^{d} \mathbf{M}^{(s)}$. If $\mathbf{M}$ is symmetric, we define $\lambda(\mathbf{M})$ to be its second largest eigenvalue (in absolute value). Following [21], if $\mathbf{M}$ is not symmetric (which is usually the case), we define $\lambda(\mathbf{M})$ as the second largest eigenvalue of the symmetric matrix $\mathbf{M} \cdot \mathbf{M}^T$, where $\mathbf{M}^T$ is the transpose of $\mathbf{M}$. We always assume that $\lambda(\mathbf{M}) < 1$, which is guaranteed to hold if the matrix $\mathbf{M}$ is irreducible. Since $\mathbf{M}$ is doubly stochastic, all powers of $\mathbf{M}$ are doubly stochastic. A natural choice for the d matching matrices is given by an edge coloring of G. There are various efficient distributed edge coloring algorithms, e.g. [20, 19].

Balancing Circuit on Specific Topologies. For *cycles*, we will consider the natural "Odd-Even" scheme meaning that for $\mathbf{M}^{(1)}$, the matching consists of all edges $\{j, (j+1) \pmod n\}$ for any odd j, while for $\mathbf{M}^{(2)}$, the matching consists of all edges $\{j, (j+1) \pmod n\}$ for any even j. More generally, for r-dimensional tori with vertex set $[0, n^{1/r} - 1]^r$, we will have $2 \cdot r$ matchings in total, meaning that for every dimension $1 \leq i \leq r$ we have two matchings along dimension i, similar to the definition of matchings for the cycle. For *hypercubes*, the canonical choice is dimension exchange consisting of $d = \log_2 n$ matching matrices $\mathbf{M}^{(i)}$ by $\mathbf{M}^{(i)}_{u,v} = 1/2$ if and only if the bit representation of u and v differ only in bit i.

Continuous Case vs. Discrete Case. In the continuous case, load is arbitrarily divisible. Let $\xi^{(0)} \in \mathbb{R}^n$ be the initial load represented as a row vector, and in every round two matched nodes average their load perfectly. We consider the load vector $\xi^{(t)}$ after t rounds in the balancing circuit model (that means, after the executions of $t \cdot d$ matchings in total). This process corresponds to a linear system $\xi^{(t)} = \xi^{(t-1)} \mathbf{M}$, which results in $\xi^{(t)} = \xi^{(0)} \mathbf{M}^t$.

Let us now turn to the discrete case with indivisible, unit-size tokens. Let $x^{(0)} \in \mathbb{N}^n$ be the initial load vector with average load $\overline{x} := \sum_{w \in V} x^{(0)}_w / n$, and $x^{(t)}$ be the load vector at the end of round t. In case the sum of tokens of the two paired nodes is odd, we employ the

[1] Note that d may be different from the maximal degree (or degree) of the underlying graph.

random orientation (or *randomized rounding*) [21, 24]. More precisely, if there are two nodes u and v with load a and b being paired by matching $\mathbf{M}^{(t)}$, then node u gets either $\lceil \frac{a+b}{2} \rceil$ or $\lfloor \frac{a+b}{2} \rfloor$ tokens, with probability $1/2$ each. The remaining tokens are assigned to node v.

The Average-Case Setting. We consider a setting where each entry of the initial load vector $x^{(0)}$ is chosen from an exponentially concentrated probability distribution D with expectation μ and variance σ^2 (see Definition 1.1). Our main results in this paper hold for all distributions satisfying the following definition.

▶ **Definition 1.1.** A distribution D over $\mathbb{N} \cup \{0\}$ with expectation μ and variance σ^2 is *exponentially concentrated* if there is a constant $\kappa > 0$ so that for any $X \sim D$, $\delta > 0$,

$$\mathbb{P}\left[|X - \mu| \geq \delta \cdot \sigma\right] \leq \exp\left(-\kappa\delta\right).$$

In the following, we refer to **average-case** when the initial number of load items on each vertex is drawn independently from a fixed exponentially concentrated distribution.

▶ **Lemma 1.2.** *The uniform distribution, binomial distribution, geometric distribution and Poisson distribution are all exponentially concentrated.*

▶ **Lemma 1.3.** *Let D be an exponentially concentrated distribution and let $X \sim D$. Then,*

$$\mathbb{P}\left[|X - \mu| \leq 8/\kappa \cdot \sigma \log n\right] \geq 1 - n^{-3}.$$

In particular, the initial discrepancy satisfies $\operatorname{disc}(x^{(0)}) = O(\sigma \cdot \log n)$ *with probability* $1 - n^{-2}$.

The advantage of Lemma 1.3 is that we can use a simple conditioning trick to work with distributions that have a finite range and can therefore be analyzed by Hoeffding's inequality. Therefore in the analysis we may simply work with a bounded-range distribution $\widetilde{D}$, which is D under the condition that only values in the interval $[\mu - 8/\kappa \cdot \sigma \log n, \mu + 8/\kappa \cdot \sigma \log n]$ occur.

1.2 Our results

Our first contribution is a general formula that allows us to express the load difference between an arbitrary pair of nodes in round t.

▶ **Theorem 1.4.** *Consider the balancing circuit model with an arbitrary round matrix $\mathbf{M}$ in the average case. Then for any pair of nodes u, v and round t, it holds for any $\delta > 0$ that*

$$\mathbb{P}\left[\left|x_u^{(t)} - x_v^{(t)}\right| \geq \delta \cdot 16\sqrt{2}/\kappa \cdot \sigma \cdot \log n \cdot \left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2 + \sqrt{64 \log n}\right] \leq 2 \cdot e^{-\delta^2} + 2n^{-3}.$$

Furthermore, for any pair of vertices u, v, we have the following lower bound:

$$\mathbb{P}\left[\left|x_u^{(t)} - x_v^{(t)}\right| \geq \sigma/(2\sqrt{2 \log_2 \sigma}) \cdot \left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2 - \sqrt{64 \log n}\right] \geq \frac{1}{16}.$$

▶ Remark. The lower bound above is useless if σ is small, say, at most a constant. However for sufficiently large σ, the lower bound gives a useful result (see also Section 4 & 5).

The proof of the upper bound Theorem 1.4 is the easier direction, and it relies on a previous result relating continuous and discrete load balancing from [24]. The lower bound is technically more challenging and applies a generalized version of the central limit theorem.

Together, the upper and lower bound in the above result establish that the load deviation between any two nodes u and v is essentially captured by $\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2$. However, in

some instances it might be desirable to have a more tangible estimate at the expense of generality. To this end, we first observe that $\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2^2 \leq 4 \cdot \max_{k\in V} \left\|\mathbf{M}^t_{\cdot,k} - \frac{\mathbf{1}}{\mathbf{n}}\right\|_2^2$ (see Lemma 3.1). Hence we are left with the problem of bounding the t-step probability vector $\mathbf{M}^t_{\cdot,k}$.

For reversible Markov chains, the last expression has been analyzed in several works. For example, [15, Lemma 3.6] implies that for random walks on graphs, $\mathbf{P}^t_{u,v} = O(\deg(v)/\sqrt{t})$. However, the Markov chain associated to $\mathbf{M}$ is not reversible in general. For irreversible Markov chains, [14] used the so-called evolving set process to derive a similar bound. Specifically, they proved in [14, Theorem 17.17] that if $\mathbf{P}$ denotes the transition matrix of a lazy random walk (i.e., a random walk with loop probability at least 1/2) on a graph with maximal degree Δ, π the stationary distribution of $\mathbf{P}$, then for any vertex $x \in V$:

$$\left|\mathbf{P}^t_{x,x} - \pi_x\right| \leq \frac{\sqrt{2}\Delta^{5/2}}{\sqrt{t}}.$$

Such estimates have been used in applications besides load balancing, including distributed random walks and spanning tree enumeration [23, 15]. Here we generalize this result to Markov chains with an arbitrary loop probability and to arbitrary t-step transition probabilities:

▶ **Theorem 1.5.** *Let $\mathbf{P}$ be the transition matrix of an irreducible Markov chain and π its stationary distribution. Then we have for all states x, y and step t,*

$$\left|\mathbf{P}^t_{x,y} - \pi_y\right| \leq \frac{\pi_{\max}^{3/2}}{\pi_{\min}^{3/2}} \cdot \frac{2}{\beta^{1/2}\alpha}\sqrt{\frac{1-\beta+\alpha}{\alpha t}},$$

where $\alpha := \min_{u\neq v} \mathbf{P}_{u,v} > 0$ and $\beta := \min_u \mathbf{P}_{u,u} > 0$.

Applying this bound to a round matrix $\mathbf{M}$ formed of $d = O(1)$ matchings we obtain $\left|\mathbf{M}^t_{u,v} - 1/n\right| = O(t^{-1/2})$. It should be noted that [24, Lemma 2.5] proved a weaker version where the upper bound is only $O(t^{-1/8})$ instead of $O(t^{-1/2})$. As proven in Lemma 4.2, the bound $O(t^{-1/2})$ is asymptotically tight if we consider the balancing circuit model on *cycles*.

Combining the bound in Theorem 1.5 with the upper bound in Theorem 1.4 yields:

▶ **Theorem 1.6.** *Consider the balancing circuit model with an arbitrary round matrix $\mathbf{M}$ consisting of $d = O(1)$ matchings in the average case. The discrepancy after t rounds is $O(t^{-1/4} \cdot \sigma \cdot (\log n)^{3/2} + \sqrt{\log n})$ with probability $1 - O(n^{-1})$.*

Since the initial discrepancy in the average case is $O(\sigma \cdot \log n)$ (see Lemma 1.3), Theorem 1.6 implies that in the average case, there is a significant decrease (roughly of order $t^{-1/4}$) in the discrepancy, regardless of the underlying topology. For round matrices $\mathbf{M}$ with small second largest eigenvalue, the next result provides a significant improvement:

▶ **Theorem 1.7.** *Consider the balancing circuit model with an arbitrary round matrix $\mathbf{M}$ consisting of d matchings in the average case. We can derive that the discrepancy after t rounds is $O(\lambda(\mathbf{M})^{t/4} \cdot \sigma \cdot (\log n)^{3/2} + \sqrt{\log n})$ with probability $1 - O(n^{-1})$.*

In Section 4, we derive bounds on the discrepancy for concrete topologies (see Table 1).

Finally, we discuss our results and compare them with the convergence of the discrepancy in the worst-case in Section 5. On a high level, these results demonstrate that on all the considered topologies, we have much faster convergence in the average-case than in the worst-case. However, if we are only interested in the time to achieve a very small, say, constant or poly-logarithmic discrepancy, then we reveal an interesting dichotomy: we have a quicker convergence than in the worst-case iff the standard deviation σ is smaller than some threshold depending on the topology. We observe the same phenomena in our experiments.

Table 1 Discrepancy bounds (without logarithmic factors) for different topologies.

Graph	$\mathrm{disc}(x^{(t)})$
Cycle	$t^{-1/4} \cdot \sigma$
r-dim. Torus	$t^{-r/4} \cdot \sigma$
Expander	$\lambda^{t/4} \cdot \sigma$
Hypercube	$2^{-t/2} \cdot \sigma$

2 Proof of the General Bound (Theorem 1.4)

2.1 Proof of Theorem 1.4 (Upper Bound)

We will use the following result from [24] that bounds the deviation between the continuous and discrete load, assuming that we have $\xi^{(0)} = x^{(0)}$.

▶ **Theorem 2.1** ([24, Theorem 3.6(i)]). *Consider the balancing circuit model with an arbitrary round matrix* $\mathbf{M}$*. Then for any round* $t \geq 1$ *it holds that*

$$\mathbb{P}\left[\max_{w \in V} \left|x_w^{(t)} - \xi_w^{(t)}\right| \leq \sqrt{16 \cdot \log n}\right] \geq 1 - 2n^{-3}.$$

The basic proof idea is as follows. Since $\xi_u^{(t)} - \xi_v^{(t)} = \sum_{w \in V} \xi_w^{(0)} \cdot (\mathbf{M}_{w,u}^t - \mathbf{M}_{w,v}^t)$, it is a weighted sum of n i.i.d. random variables and its expectation is 0. We then can apply Hoeffding's inequality to obtain the theorem.

2.2 Proof of Theorem 1.4 (Lower Bound)

The proof of the lower bound will use the following quantitative version of a central limit type theorem for independent but non-identical random variables.

▶ **Theorem 2.2** (Berry-Esseen [5, 10] for non-identical r.v.). *Let* $X_1, X_2, ..., X_n$ *be independently distributed with* $\mathbb{E}[X_i] = 0$, $\mathbb{E}\left[X_i^2\right] = \mathrm{Var}(X_i) = \sigma_i^2$, *and* $\mathbb{E}\left[|X_i|^3\right] = \rho_i < \infty$. *If* $F_n(x)$ *is the distribution of* $\frac{X_1 + ... + X_n}{\sqrt{\sigma_1^2 + \sigma_2^2 + ... + \sigma_n^2}}$ *and* $\Phi(x)$ *is the standard normal distribution, then*

$$|F_n(x) - \Phi(x)| \leq C_0 \psi_0,$$

where $\psi_0 = \left(\sum_{i=1}^n \sigma_i^2\right)^{-3/2} \cdot \sum_{i=1}^n \rho_i$ *and* $C_0 > 0$ *is a constant.*

With this concentration tool at hand, we are able to prove the lower bound in Theorem 1.4. Unfortunately, it appears quite difficult to apply Theorem 2.2 directly to $\xi_u^{(t)} - \xi_v^{(t)}$, since we need a good bound on the error term ψ_0. To this end, we will first partition the vertex set V into buckets with equal contribution to $\xi_u^{(t)} - \xi_v^{(t)}$. Then we will apply Theorem 2.2 to the bucket with the largest variance.

Proof of Theorem 1.4 (Lower Bound). We first consider $\xi_u^{(t)} - \xi_v^{(t)}$:

$$dev := \xi_u^{(t)} - \xi_v^{(t)} = \sum_{w \in V} \xi_w^{(0)} \cdot \left(\mathbf{M}_{w,u}^t - \mathbf{M}_{w,v}^t\right),$$

which is a weighted sum of i.i.d. random variables with expectation μ and variance σ^2. As mentioned earlier, we have $\mathbb{E}[dev] = \sum_{w \in V} \mathbb{E}\left[\xi_w^{(0)}\right] \cdot \left(\mathbf{M}_{w,u}^t - \mathbf{M}_{w,v}^t\right) = 0$ since $\mathbf{M}$ is a doubly stochastic matrix. Of course, we could apply Theorem 2.2 directly to dev, but it

appears difficult to control the error term ψ_0. Therefore we will first partition the above sum into buckets where the weights of the random variables are roughly the same.

More precisely, we will partition V into $2\log_2\sigma$ buckets, where for each i we have $V_i := \left\{w \in V : |\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}| \in (2^{-i-1}, 2^{-i}]\right\}$ for $1 \le i \le 2\log_2\sigma - 1$, and $V_{2\log_2\sigma} := \left\{w \in V : \left|\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right| \le \frac{1}{\sigma^2}\right\}$.

Further, let us consider the variance of dev, since $\mathrm{Var}(aX) = a^2\mathrm{Var}(X)$ and the inputs are independent random variables:

$$\sigma^2_{dev} = \sum_{w\in V} \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2 \sigma^2.$$

Then by the pigeonhole principle there exists an index $1 \le i \le 2\log_2\sigma$ such that

$$\sum_{w\in V_i} \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2 \sigma^2 \ge \frac{1}{2\log_2\sigma} \cdot \sigma^2_{dev}.$$

This is equivalent to

$$\sum_{w\in V_i} \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2 \ge \frac{1}{2\log_2\sigma} \cdot \sum_{w\in V} \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2.$$

Firstly, if such index i is just $2\log_2\sigma$, we can prove the lower bound easily. Now we can derive that, for all w in V_i,

$$\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2^2 \le \left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_\infty \cdot \left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_1 \le \frac{1}{\sigma^2} \cdot 2 = O(\sigma^{-2}).$$

Then in Theorem 1.4 $\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2$ is $O(\sigma^{-1})$ and the lower bound holds trivially. Therefore, we will assume in the remainder of the proof that $i < 2\log_2\sigma$.

We now decompose dev into $dev = S + S^c$, where

$$S := \sum_{w\in V_i} \xi^{(0)}_w \cdot \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right) \text{ and } S^c := \sum_{w\notin V_i} \xi^{(0)}_w \cdot \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right).$$

Let us first analyze S. We need to apply Theorem 2.2 to S. Before applying Theorem 2.2, we scale the original distribution to $\xi^{'(0)}_w = \xi^{(0)}_w - \mu$ to make the expectation be 0. In preparation for this, let us first upper bound ψ_0. Using the definition of exponentially concentrated distributions, it follows that for any constant k, the first k moments of $\xi^{'(0)}_w$ are all bounded from above by $O(\sigma^k)$. Hence,

$$\psi_0 = \frac{\sum_{w\in V_i} \mathbb{E}\left[\left|\xi^{'(0)}_w \cdot \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)\right|^3\right]}{\left(\sum_{w\in V_i} \mathbb{E}\left[\left(\xi^{'(0)}_w \cdot \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)\right)^2\right]\right)^{3/2}} \le \frac{O(\sigma^3) \cdot \sum_{w\in V_i} \left|\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right|^3}{\sigma^3 \left(\sum_{w\in V_i} \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2\right)^{3/2}}.$$

Recalling that for any $w \in V_i$, $\left|\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right| \in (2^{-i-1}, 2^{-i}]$, we can simplify the above expression as follows:

$$\psi_0 = O\left(\frac{|V_i| \cdot 2^{-3i}}{|V_i|^{3/2} \cdot 2^{-3i}}\right) = O(|V_i|^{-1/2}).$$

In the following, we will assume that $|V_i| \geq C_1$, where $C_1 > 0$ is a sufficiently large constant to be specified later. Since $\mathrm{Var}(aX) = a^2\mathrm{Var}(X)$, we have

$$F_n(x) = \mathbb{P}\left[\frac{\sum_{w\in V_i} \xi_w^{'(0)} \cdot \left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)}{\sigma\sqrt{\sum_{w\in V_i}\left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2}} \leq x\right]$$
$$= \mathbb{P}\left[S - \mathbb{E}\left[S\right] \leq x\sigma\sqrt{\sum_{w\in V_i}\left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2}\right].$$

Since $|V_i| \geq C_1$, there is a constant $C_2 = C_2(C_1, C_0) > 0$ such that $C_0 \cdot \psi_0 \leq C_2$ and

$$\mathbb{P}\left[S - \mathbb{E}\left[S\right] \geq x\sigma\sqrt{\sum_{w\in V_i}\left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2}\right] \geq \Phi(-x) - C_0\psi_0 \geq \Phi(-x) - C_2.$$

Now let $\Phi^c(x)$ denote the complement of the standard normal distribution. By using [1, Formula 7.1.13] and substitution we get:

$$\frac{1}{\sqrt{\pi}(x + \sqrt{x^2+2})e^{x^2}} < \Phi^c(x) \leqslant \frac{1}{\sqrt{\pi}(x + \sqrt{x^2+4/\pi})e^{x^2}}.$$

Hence by $\Phi(-x) = \Phi^c(x)$, choosing $x = 1$ and C_1 sufficiently large,

$$\mathbb{P}\left[S - \mathbb{E}\left[S\right] \geq \sigma\sqrt{\sum_{w\in V_i}\left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2}\right] \geq \frac{1}{16}.$$

Similarly, we can derive that

$$\mathbb{P}\left[\mathbb{E}\left[S\right] - S \geq \sigma\sqrt{\sum_{w\in V_i}\left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2}\right] \geq \frac{1}{16}.$$

Hence, independent of what the value S^c is, there is still a probability of at least $1/16$ so that $|S + S^c| \geq \sigma/2 \cdot \sqrt{1/(2\log_2\sigma)} \cdot \sqrt{\sum_{w\in V}\left(\mathbf{M}^t_{w,u} - \mathbf{M}^t_{w,v}\right)^2}$, which completes the proof for the case $|V_i| \geq C_1$. The case $|V_i| < C_1$ is similar and omitted here. The basic idea is not to apply Berry-Esseen but simply use the fact that any exponentially distributed random variable deviates from the expectation by $\Omega(\sigma)$ with constant probability. ◀

3 Proof of the Universal Bounds (Theorem 1.6, Theorem 1.7)

In the previous section we proved that the deviation between the loads of two nodes u and v is essentially captured by $\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2$. However, in some cases it might be hard to compute or estimate this quantity for arbitrary vertices u and v. Therefore we will establish Theorem 1.6 which gives a more concrete estimate.

3.1 Proof of Theorem 1.6

The proof of Theorem 1.6 is fairly involved and we sketch the high level ideas. We first show that $\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2^2$ can be upper bounded in terms of the ℓ_2-distance to the stationary distribution.

▶ **Lemma 3.1.** *Consider the balancing circuit model with an arbitrary round matrix* $\mathbf{M}$*. Then for all* $u, v \in V$*, we have* $\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\|_2^2 \leq 4 \cdot \max_{k\in V} \|\mathbf{M}^t_{\cdot,k} - \frac{\mathbf{1}}{\mathbf{n}}\|_2^2$*. Further, for any* $u \in V$ *we have* $\max_{v\in V} \|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\|_2^2 \geq \|\mathbf{M}^t_{\cdot,u} - \frac{\mathbf{1}}{\mathbf{n}}\|_2^2$*.*

The next step and main ingredient of the proof of Theorem 1.6 is to establish that $\|\mathbf{M}^t_{\cdot,k} - \frac{\mathbf{1}}{\mathbf{n}}\|_\infty = O(1/\sqrt{t})$. This result will be a direct application of a general bound on the t-step probabilities of an arbitrary, possibly non-reversible Markov chain, as given in Theorem 1.5 from page 5:

In this subsection we prove Theorem 1.6, assuming the correctness of Theorem 1.5 whose proof is deferred to Section 3.2.

Proof of Theorem 1.6. By Theorem 1.4 and Lemma 3.1, we obtain

$$\mathbb{P}\left[\left|x_u^{(t)} - x_v^{(t)}\right| \geq \delta \cdot 16\sqrt{2}/\kappa \cdot \sigma \cdot \log n \cdot \max_{k\in V}\left\|\mathbf{M}^t_{\cdot,k} - \frac{\mathbf{1}}{\mathbf{n}}\right\|_2 + \sqrt{64 \log n}\right] \leq 2e^{-\delta^2} + 2n^{-3}.$$

Hence we can find a $\delta = \sqrt{3 \log n}$ so that the latter probability gets smaller than $4n^{-3}$. Further, by applying Theorem 1.5 with $\alpha = \beta = 2^{-d}$ to $\mathbf{P} = \mathbf{M}$ we conclude that $\|\mathbf{M}^t_{\cdot,k} - \frac{\mathbf{1}}{\mathbf{n}}\|_\infty = O(t^{-1/2})$, since $d = O(1)$. Using $\|.\|_2^2 \leq \|.\|_\infty \cdot \|.\|_1$, $\|\mathbf{M}^t_{\cdot,k} - \frac{\mathbf{1}}{\mathbf{n}}\|_2^2 = O(t^{-1/2})$ and the union bound, $\mathrm{disc}(x^{(t)}) = O(t^{-1/4} \cdot \sigma \cdot (\log n)^{3/2} + \sqrt{\log n})$ with probability at least $1 - 4n^{-1}$. ◀

3.2 Proof of Theorem 1.5

This section is devoted to the proof of Theorem 1.5. Our proof is based on the evolving-set process, which is a Markov chain based on any given irreducible, not necessarily reversible Markov chain on Ω. For the definition of the evolving set process, we closely follow the exposition in [14, Chapter 17].

Let $\mathbf{P}$ denote the transition matrix of an irreducible Markov chain and π its stationary distribution. $\mathbf{P}^t$ is the t-step transition probability matrix. The *edge measure* Q is defined by $Q_{x,y} := \pi_x \mathbf{P}_{x,y}$ and $Q(A, B) = \sum_{x\in A, y\in B} Q_{x,y}$.

▶ **Definition 3.2.** Given a transition matrix $\mathbf{P}$, the **evolving-set process** is a Markov chain on subsets of Ω defined as follows. Suppose the current state is $S \subset \Omega$. Let U be a random variable which is uniform on $[0, 1]$. The next state of the chain is the set

$$\tilde{S} = \left\{y \in \Omega : \frac{Q(S, y)}{\pi_y} \geq U\right\}.$$

This chain is *not irreducible* because $\varnothing$ and Ω are absorbing states. It follows that

$$\mathbb{P}\left[y \in S_{t+1} \,|\, S_t\right] = \frac{Q(S_t, y)}{\pi_y}$$

since the probability that $y \in S_{t+1}$ is equal to the probability of the event that the chosen value of U is less than $\frac{Q(S_t,y)}{\pi_y}$.

▶ **Proposition 3.3** ([14, Proposition 17.19])**.** *Let* (M_t) *be a non-negative martingale with respect to* (Y_t)*, and define* $T_h := \min\{t \geq 0 : M_t = 0 \text{ or } M_t \geq h\}$*. Assume that for any* $h \geq 0$

- *For* $t < T_h$*,* $\mathrm{Var}(M_{t+1} \,|\, Y_0, \ldots, Y_t) \geq \sigma^2$*, and*
- $M_{T_h} \leq Dh$*.*

Let $T := T_1$*. If* M_0 *is a constant, then* $\mathbb{P}\left[T > t\right] \leq \frac{2M_0}{\sigma}\sqrt{\frac{D}{t}}$*.*

We now generalize [14, Lemma 17.14] to cover arbitrarily small loop probabilities.

▶ **Lemma 3.4.** *Let (U_t) be a sequence of independent random variables, each uniform on $[0,1]$, such that S_{t+1} is generated from S_t using U_{t+1}. Then with $\beta := \min_u \mathbf{P}_{u,u} > 0$,*

$$\mathbb{E}\left[\pi(S_{t+1}) \,|\, U_{t+1} \le \beta, S_t = S\right] \ge \pi(S) + Q(S, S^c),$$
$$\mathbb{E}\left[\pi(S_{t+1}) \,|\, U_{t+1} > \beta, S_t = S\right] \le \pi(S) - \frac{\beta Q(S, S^c)}{1-\beta}.$$

The derivation of the next lemma closely follows the analysis in [14, Chapter 17].

▶ **Lemma 3.5.** *For any two states x, y, $\left|\mathbf{P}^t_{x,y} - \pi_y\right| \le \frac{\pi_y}{\pi_x} \cdot \mathbb{P}_{\{x\}}\left[\tau > t\right]$.*

Now we want to use Proposition 3.3 to bound $\mathbb{P}_{\{x\}}[\tau > t]$. To apply it, we substitute the following parameters: M_0 is be $\pi(\{x\})$, Y_t is S_t, and $T = T_1 := \min\{t \ge 0 : \pi(S_t) = 0 \text{ or } \pi(S_t) \ge 1\}$. Hence in our case, τ is the same as T (or T_1) in the proposition. The following two lemmas elaborate on the two preconditions (i) and (ii) of Proposition 3.3.

▶ **Lemma 3.6.** *For any time t and $S_0 = \{x\}$, $\mathrm{Var}_{S_t}(\pi(S_{t+1})) \ge \beta \pi_{\min}^2 \alpha^2$.*

Finally, we derive an upper bound on the amount by which S_t can increase in one iteration.

▶ **Lemma 3.7.** *For any time t and $S_0 = \{x\}$, $\pi(S_{t+1}) \le \left(\frac{1-\beta}{\alpha} + 1\right) \frac{\pi_{\max}}{\pi_{\min}} \cdot \pi(S_t)$.*

The proof of Theorem 1.5 follows then by combining Proposition 3.3, Lemma 3.4, Lemma 3.5, Lemma 3.6 and Lemma 3.7.

3.3 Proof of Theorem 1.7

We now prove the following discrepancy bound that depends on the $\lambda(\mathbf{M})$, as defined in Section 1.1.

Proof of Theorem 1.7. By [24, Lemma 2.4], for any pair of vertices $u, v \in V$, $\left|\mathbf{M}^t_{u,v} - \frac{1}{n}\right| \le \lambda(\mathbf{M})^{t/2}$. Hence by Lemma 3.1, $\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2 = O(\lambda(\mathbf{M})^{t/4})$. The bound on the discrepancy follows from Theorem 1.4 and the union bound over all vertices. ◀

4 Applications to Different Graph Topologies

Cycles. Recall that for the cycle, $V = \{0, \ldots, n-1\}$ is the set of vertices, and the distance between two vertices is $\mathrm{dist}(x,y) = \min\{y - x, x + n - y\}$ for any pair of vertices $x < y$.

The upper bound on the discrepancy follows directly from Theorem 1.6, and it only remains to prove the lower bound. To this end, we will apply the lower bound in Theorem 1.4 and need to derive a lower bound on $\|\mathbf{M}^t_{\cdot,u} - \frac{\mathbf{1}}{\mathbf{n}}\|_2^2$. Intuitively, if we had a simple random walk, we could immediately infer that this quantity is $\Omega(1/\sqrt{t})$. Since after t steps, the random walk is with probability $\approx 1/\sqrt{t}$ at any vertex with distance at most $O(\sqrt{t})$. To prove that this also holds for the load balancing process, we first derive a concentration inequality that upper bounds the probability for the random walk to reach a distant state:

▶ **Lemma 4.1.** *Consider the standard balancing circuit model on the cycle with round matrix $\mathbf{M}$. Then for any $u \in V$ and $\delta \in (0, n/2 - 1)$, we have*

$$\sum_{v \in V:\ \mathrm{dist}(u,v) \ge \delta} \mathbf{M}^t_{u,v} \le 2 \cdot \exp\left(-\frac{(\delta-2)^2}{8t}\right).$$

With the help of Lemma 4.1, we can indeed verify our intuition:

▶ **Lemma 4.2.** *Consider the standard balancing circuit model on the cycle with round matrix* $\mathbf{M}$*. Then for any vertex* $u \in V$*,* $\|\mathbf{M}^t_{\cdot,u} - \frac{\mathbf{1}}{\mathbf{n}}\|_2^2 = \Omega(1/\sqrt{t})$.

Lemma 4.2 also proves that the factor $\sqrt{1/t}$ in the upper bound in Theorem 1.5 is the best possible. The lower bound on the discrepancy now follows by combining Lemma 4.2 with Theorem 1.4 and Lemma 3.1 stating that for any vertex $u \in V$, there exists another vertex $v \in V$ such that $\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\|_2^2 \geq \|\mathbf{M}^t_{\cdot,u} - \frac{\mathbf{1}}{\mathbf{n}}\|_2^2 = \Omega(1/\sqrt{t})$.

Tori. In this section we consider r-dimensional tori, where $r \geq 1$ is any constant. For the upper bound, note that the computation of $\mathbf{M}^t_{\cdot,\cdot}$ can be decomposed to independent computations in the r dimensions, and each dimension has the same distribution as the cycle on $n^{1/r}$ vertices. Specifically, if we denote by $\widetilde{\mathbf{M}}$ the round matrix of the standard balancing circuit scheme on the cycle with $n^{1/r}$ vertices and $\mathbf{M}$ is the round matrix of the r-dimensional torus with n vertices, then for any pair of vertices $x = (x_1, \ldots, x_r), v = (y_1, \ldots, y_r)$ on the torus we have $\mathbf{M}^t_{x,y} = \prod_{i=1}^r \widetilde{\mathbf{M}}^t_{x_i,y_i}$. From Theorem 1.5, $|\widetilde{\mathbf{M}}^t_{x_i,y_i} - \frac{1}{n^{1/r}}| = O(t^{-1/2})$, and therefore, since r is a constant,

$$\mathbf{M}^t_{x,y} \leq \prod_{i=1}^{r} \left(\frac{1}{n^{1/r}} + O(t^{-1/2}) \right) = O(t^{-r/2} + n^{-1}),$$

and thus $\left\|\mathbf{M}^t_{x,y} - \frac{\mathbf{1}}{\mathbf{n}}\right\|_2^2 = O(t^{-r/2})$ for any pair of vertices x, y. Hence by Lemma 3.1, $\left\|\mathbf{M}^t_{\cdot,u} - \mathbf{M}^t_{\cdot,v}\right\|_2^2 = O(t^{-r/2})$. Plugging this bound into Theorem 1.4 yields that the load difference between any pair of the nodes u and v at round t is at most $O(t^{-r/4} \cdot \sigma \cdot \log^{3/2} n + \sqrt{\log n})$ with probability at least $1 - 4n^{-1}$. The bound on the discrepancy now simply follows by the union bound.

We now turn to the lower bound on the discrepancy. With the same derivation as in Lemma 4.2 we obtain the following result:

▶ **Lemma 4.3.** *Consider the standard balancing circuit model on the* r*-dimensional torus with round matrix* $\mathbf{M}$*. Then for any vertex* $u \in V$*,* $\|\mathbf{M}^t_{\cdot,u} - \frac{\mathbf{1}}{\mathbf{n}}\|_2^2 = \Omega(t^{-r/2})$.

The lower bound on the torus follows by combining Lemma 4.3 and Theorem 1.4.

Expanders. The upper bound $O(\lambda(\mathbf{M})^{t/4} \cdot \sigma \cdot (\log n)^{3/2} + \sqrt{\log n})$ for expanders follows immediately from Theorem 1.7. For the lower bound, since the round matrix consists of d matchings, it is easy to verify that whenever $\mathbf{M}^t_{u,v} > 0$, we have $\mathbf{M}^t_{u,v} \geq 2^{-d \cdot t}$. Consequently, for any vertex $u \in V$, $\left\|\mathbf{M}^t_{\cdot,u} - \frac{\mathbf{1}}{\mathbf{n}}\right\|_2^2 = \Omega(2^{-d \cdot t})$. Plugging this into Theorem 1.4 yields a lower bound on the discrepancy which is $\Omega(2^{-d \cdot t/2} \cdot \sigma / \sqrt{\log \sigma})$.

Hypercubes. For the hypercube, there is a worst-case bound of $\log_2 \log_2 n + O(1)$ [16, Theorem 5.1 & 5.3] for any input after $\log_2 n$ iterations of the dimension-exchange, i.e., after one execution of the round matrix. Hence, we will only analyze the discrepancy after s matchings, where $1 \leq s < \log_2 n$. By applying the same analysis as in Theorem 1.7, but now with $|\prod_{s=1}^t \mathbf{M}^{(s)}_{u,v} - \frac{1}{n}| \leq 2^{-t}$, we obtain that the discrepancy is $O(2^{-t/2} \cdot \sigma \cdot (\log n)^{3/2} + \sqrt{\log n})$. Applying Theorem 1.4, we obtain the lower bound $\Omega(2^{-t/2} \cdot \sigma / \sqrt{\log \sigma})$.

5 Discussion and Empirical Results

We will now compare our average-case to a worst-case scenario on cycles, 2D-tori and hypercubes. For the sake of concreteness, we always assume that the input is drawn from a

uniform distribution $\mathsf{Uni}[0, 2K]$. Our choice for the worst-case load vector will have (roughly) the same number of tokens and initial discrepancy. However, the exact definition of the vector will depend on the underlying topology.

Cycles and 2D-Tori For the worst-case setting on cycles, fix an arbitrary node $u \in V$ and let all nodes with distance at most $n/4$ initially have a load of $2K$ while all the other nodes have load 0. This gives rise to a load vector with nK tokens and initial discrepancy $2K$. For 2D-tori, fix an arbitrary node $u \in V$ and assign a load of $2K$ to the $n/2$ nearest neighbors and load 0 otherwise. This defines a load vector with nK tokens and initial discrepancy $2K$.

The next result provides a lower bound on the discrepancy for cycles and 2D-tori in the aforementioned worst-case setting. It essentially shows that for worst-case inputs, $\Omega(n^2)$ rounds and $\Omega(n)$ rounds are necessary for the cycle, 2D-tori, respectively, in order to reduce the discrepancy by more than a constant factor. This stands in sharp contrast to Theorem 1.6, proving a decay of the discrepancy by $\approx t^{-1/4}$, starting from the first round.

▶ **Proposition 5.1.** *For the aforementioned worst-case setting on the cycle, it holds for any round $t > 0$ that* $\operatorname{disc}(x^{(t)}) \geq \frac{1}{8} \cdot K \cdot \left(1 - \exp\left(-\frac{n^2}{2048t}\right)\right) - \sqrt{64 \log n}$, *with probability at least $1 - n^{-1}$. Further, for 2D-tori, it holds for any round $t > 0$ that* $\operatorname{disc}(x^{(t)}) \geq \frac{1}{8} \cdot K \cdot \left(1 - \exp\left(-\frac{n}{2048t}\right)\right) - \sqrt{64 \log n}$, *with probability at least $1 - n^{-1}$.*

Hypercube. We will consider only $\log_2 n$ rounds, since the discrepancy is $\log_2 \log_2 n + O(1)$ after $\log_2 n$ rounds and $O(1)$ after $2 \log_2 n$ rounds [16]. A natural corresponding worst-case distribution is to have load $2K$ on all nodes whose $\log_2 n$-th bit is equal to 1 and 0 otherwise. This way the discrepancy is only reduced in the final round $\log_2 n$.

Experiments. For each of the cycle, 2D-torus and hypercube, we consider two comparative experiments with an average-case initial load vector and a worst-case initial load vector.

The first experiment considers a "lightly loaded case", where the theoretical results suggest that a small (i.e., constant or logarithmic) discrepancy is reached before the expected worst-case load balancing time, which are $\approx n^2$ for cycles and $\approx n$ for 2D-tori. The second experiment considers a "heavily loaded case", where the theoretical results suggest that a small discrepancy is not reached faster than in the worst-case.

For cycles and 2D-tori we choose for the lightly loaded case $K = \sqrt{n}$ and for the heavily loaded case $K = n^2$. The experiments confirm the theoretical results in the sense that for both choices of K, we have a much quicker convergence of the discrepancy than in the worst case. However, the experiments also demonstrate that only in the lightly loaded case we reach a small discrepancy quickly, whereas in the heavily loaded case there is no big difference between worst-case and average-case if it comes to the time to reach a small discrepancy.

On the hypercube, our bounds on the discrepancy suggest a smaller K in comparison to the experiments on cycles and 2D-tori. That is why we choose $K = n^{1/4}$ in the lightly loaded case and $K = n$ in the heavily loaded case. With these adjustments of K in both cases, the experimental results of the hypercube are inline with the ones for the cycle and 2D-tori. More details including the plots can be found in the full version [8].

Acknowledgements. We are grateful to the reviewers for their valuable comments. We are also thankful to Robert Elsässer and Yuchen Yang for helpful discussions.

References

1. Milton Abramowitz, Irene A Stegun, et al. Handbook of mathematical functions. *Applied mathematics series*, 55:62, 1966.
2. Dan Alistarh, Keren Censor-Hillel, and Nir Shavit. Are lock-free concurrent algorithms practically wait-free? *J. ACM*, 63(4):31:1–31:20, 2016.
3. Aris Anagnostopoulos, Adam Kirsch, and Eli Upfal. Load balancing in arbitrary network topologies with stochastic adversarial input. *SIAM J. Comput.*, 34(3):616–639, 2005.
4. Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Find your place: Simple distributed algorithms for community detection. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 940–959, 2017.
5. Andrew C Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–136, 1941.
6. J. E. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Pract. Exper.*, 2:289–313, 1990.
7. S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized Gossip Algorithms. *IEEE Transactions on Information Theory and IEEE/ACM Transactions on Networking*, 52:2508–2530, 2006.
8. Leran Cai and Thomas Sauerwald. Randomized load balancing on networks with stochastic inputs, 2017. `arXiv:1703.08702`.
9. G. Cybenko. Load balancing for distributed memory multiprocessors. *J. Parallel and Distributed Comput.*, 7:279–301, 1989.
10. Carl-Gustaf Esseen. *On the Liapounoff limit of error in the theory of probability*. Almqvist & Wiksell, 1942.
11. Bhaskar Ghosh, Frank Thomson Leighton, Bruce M. Maggs, S. Muthukrishnan, C. Greg Plaxton, Rajmohan Rajaraman, Andréa W. Richa, Robert Endre Tarjan, and David Zuckerman. Tight analyses of two local load balancing algorithms. *SIAM Journal on Computing*, 29(1):29–64, 1999.
12. Ashish Goel and Piotr Indyk. Stochastic load balancing and related problems. In *40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 579–586, 1999.
13. Kenneth H. Huebner, Donald L. Dewhirst, Douglas E. Smith, and Ted G. Byrom. *The Finite Element Methods for Engineers*. Wiley, 2001.
14. David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009.
15. Russell Lyons. Asymptotic enumeration of spanning trees. *Combinatorics, Probability & Computing*, 14(4):491–522, 2005.
16. Marios Mavronicolas and Thomas Sauerwald. The impact of randomization in smoothing networks. *Distributed Computing*, 22(5-6):381–411, 2010.
17. S. Muthukrishnan and Bhaskar Ghosh. Dynamic load balancing by random matchings. *J. Comput. Syst. Sci.*, 53:357–370, 1996.
18. S. Muthukrishnan, Bhaskar Ghosh, and Martin H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory Comput. Syst.*, 31:331–354, 1998.
19. A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.
20. Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. 24th Symp. Theory of Computing (STOC)*, pages 581–592, 1992.

21 Yuval Rabani, Alistair Sinclair, and Rolf Wanka. Local divergence of Markov chains and the analysis of iterative load balancing schemes. In *Proc. 39th Symp. Foundations of Computer Science (FOCS)*, pages 694–705, 1998.

22 Peter Sanders. Analysis of nearest neighbor load balancing algorithms for random loads. *Parallel Computing*, 25(8):1013–1033, 1999.

23 Atish Das Sarma, Danupon Nanongkai, Gopal Pandurangan, and Prasad Tetali. Distributed random walks. *J. ACM*, 60(1):2:1–2:31, 2013. doi:10.1145/2432622.2432624.

24 Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *Proc. 53rd Symp. Foundations of Computer Science (FOCS)*, pages 341–350, 2012.

25 A.J. Sinclair and M.R. Jerrum. Approximate counting, uninform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.

26 Raghu Subramanian and Isaac D. Scherson. An analysis of diffusive load-balancing. In *Proc. 6th Symp. Parallelism in Algorithms and Architectures (SPAA)*, pages 220–225, 1994.

27 Sonesh Surana, Brighten Godfrey, Karthik Lakshminarayanan, Richard Karp, and Ion Stoica. Load balancing in dynamic structured peer-to-peer systems. *Performance Evaluation*, 63(3):217–240, 2006.

28 Roy D. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience*, 3(5):457–481, 1991.

29 Dongliang Zhanga, Changjun Jianga, and Shu Li. A fast adaptive load balancing method for parallel particle-based simulations. *Simulation Modelling Practice and Theory*, 17(6):1032–1042, 2009.

Opinion Dynamics in Networks: Convergence, Stability and Lack of Explosion*†

Tung Mai[1], Ioannis Panageas[2], and Vijay V. Vazirani[3]

1 **Georgia Institute of Technology, Atlanta, GA, USA**
tung.mai@cc.gatech.edu
2 **MIT, Cambridge, MA, USA; and**
Singapore University of Technology and Design (SUTD), Singapore
ioannisp@mit.edu
3 **Georgia Institute of Technology, Atlanta, GA, USA**
vazirani@cc.gatech.edu

Abstract

Inspired by the work of Kempe et al. [Kempe, Kleinberg, Oren, Slivkins, EC 2013], we introduce and analyze a model on opinion formation; the update rule of our dynamics is a simplified version of that of [Kempe, Kleinberg, Oren, Slivkins, EC 2013]. We assume that the population is partitioned into types whose interaction pattern is specified by a graph. Interaction leads to population mass moving from types of smaller mass to those of bigger mass. We show that starting uniformly at random over all population vectors on the simplex, our dynamics converges point-wise with probability one to an independent set. This settles an open problem of [Kempe, Kleinberg, Oren, Slivkins, EC 2013], as applicable to our dynamics. We believe that our techniques can be used to settle the open problem for the Kempe et al. dynamics as well.

Next, we extend the model of Kempe et al. by introducing the notion of birth and death of types, with the interaction graph evolving appropriately. Birth of types is determined by a Bernoulli process and types die when their population mass is less than ϵ (a parameter). We show that if the births are infrequent, then there are long periods of "stability" in which there is no population mass that moves. Finally we show that even if births are frequent and "stability" is not attained, the total number of types does not explode: it remains logarithmic in $1/\epsilon$.

1998 ACM Subject Classification G.1.5 [Numerical Analysis] Roots of Nonlinear Equations, Convergence, G.2.2 [Discrete Mathematics] Graph Theory, Network Problems

Keywords and phrases Opinion Dynamics, Convergence, Jacobian, Center-stable Manifold

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.140

1 Introduction

The birth, growth and death of political parties, organizations, social communities and product adoption groups (e.g., whether to use Windows, Mac OS or Linux) often follows common patterns, leading to the belief that the dynamics underlying these processes has much in common. Understanding this commonality is important for the purposes of predictability and hence has been the subject of study in mathematical social science for many years [4, 7, 8, 14, 26]. In recent years, the growth of social communities on the Internet, and their increasing economic and social value, has provided fresh impetus to this study [1, 2, 5, 17].

* The full version of this paper is available at https://arxiv.org/abs/1607.03881.

† Tung Mai and Vijay V. Vazirani would like to acknowledge NSF Grant CCF-1216019. Ioannis Panageas would like to acknowledge a MIT-SUTD postdoctoral fellowship.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 140; pp. 140:1–140:14

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

In this paper, we continue along these lines by building on a natural model proposed by Kempe et al. [15]. Their model consists of an influence graph G on n vertices (types, parties) into which the entire population mass is partitioned. Their main tenet is that individuals in smaller parties tend to get influenced by those in bigger parties [1]. Individuals in the two vertices connected by an edge can interact with each other. These interactions result in individuals moving from smaller to bigger in population vertices. Kempe et al. characterize stable equilibria of this dynamics via the notion of Lyapunov stability, and they show that under any stable equilibrium, the entire mass lies in an independent set, i.e., the population breaks into non-interacting islands. The message of this result is clear: a population is (Lyapunov) stable, in the sense that the system does not change by much under small perturbations, only if people of different opinions do not interact. They also showed convergence to a fixed point, not necessarily an independent set, starting from any initial population vector and influence graph. One of their main open problems was to determine whether starting uniformly at random over all population vectors on the unit simplex, their dynamics converge with probability one to an independent set.

We first settle this open problem in the affirmative for a modification of the dynamics, which however is similar to that of Kempe et al. in spirit in that it moves mass from smaller to bigger parties (the dynamics is defined below along with a justification). We believe that the ideas behind our analysis can be used to settle the open problem for the dynamics of Kempe et al. as well, via a more complicated spectral analysis of the Jacobian of the update rule of the dynamics (see Section 3.2).

Whereas the model of Kempe et al. captures and studies the effects of migration of individuals across types in a very satisfactory manner, it is quite limited in that it does not include the birth and death of types. In this paper, we model birth and death of types. In order to arrive at realistic definitions of these notions, we first conducted case studies of political parties in several countries. We present below a case study on Greek politics, but similar phenomena arise in India, Spain, Italy and Holland (see Wikipedia pages).

The Siriza party in Greece provides an excellent example of birth of a party (this information is readily available in Wikipedia pages). This party was essentially in a dormant state until the first 2012 elections in which it got 16.8% of the vote, mostly taken away from the Pasok party, which dropped from 43.9% to 13.2% in the process (Wikipedia). In the second election in 2012, Siriza increased its vote to 26.9% and Pasok dropped to 12.3%. Finally, in 2015, Siriza increased to 36.3% and Pasok dropped further to 4.7%. Another party, Potami, was formed in 2015 and got 6.1% of the vote, again mainly from Pasok. However, in a major 2016 poll, it seems to have collapsed and is likely be absorbed by other parties. In contrast, the KKE party in Greece, which had almost no interactions with the rest of the parties (and was like a disconnected component), has remained between 4.5-8.5% of the vote over the last 26 years.

Motivated by these examples, we have modeled birth and death of types in the following manner. We model population as a continuum, as is standard in population dynamics, and time is discrete. This is the same as arXiv Version 1 of [15], which is what we will refer to

[1] Changes in the sizes of political parties and other organizations can occur for a multitude of possible reasons, such as changes in economic conditions, immigration flows, wars and terrorism, and drastic changes in technology (such as the introduction of the Internet, smart phones and social media). Studying changes due to these multitude of reasons in a systematic quantitative manner is unrealistic. For this reason, many authors in computer science and the social sciences have limited their work to studying the effects of relative sizes of the groups, in itself a key factor, e.g. see [15] and references therein. Following these works, our paper also takes a similar approach.

throughout this paper; the later versions study the continuous time analog. The birth of a new type in our model is determined by a Bernoulli process, with parameter p. The newly born type absorbs mass from all other types via a randomized process given by an arbitrary distribution with finite support (see Section 2.2). After birth, the new type is connected to an arbitrary, though non-empty, set of other types. Our model has a parameter ϵ, and when the size of a type drops below ϵ, it simply dies, moving its mass equally among its neighbors.

Our rule for migration of mass, which is somewhat different from that of Kempe et al. is motivated by the following considerations. For a type u, $\boldsymbol{x}_u$ will denote the fraction of population that is of type u. Assume that types u and v have an edge, i.e., their populations interact. If so, we will assume that some individuals of the smaller type get influenced by the larger one and move to the larger one. The question is what is a reasonable assumption on the population mass that moves.

For arriving at the rule proposed in this paper, consider three situations. If $\boldsymbol{x}_u = .02$ and $\boldsymbol{x}_v = .25$, i.e., the smaller type is very small, then clearly not many people will move. If $\boldsymbol{x}_u = .22$ and $\boldsymbol{x}_v = .25$, i.e., the types are approximately of the same population size, then again we expect not many people to move. Finally, if $\boldsymbol{x}_u = .15$ and $\boldsymbol{x}_v = .25$, i.e., both types are reasonably big and their difference is also reasonably big, then we expect several people to move from the smaller to the bigger type. From these considerations, we propose that the amount of population mass moving from v to u, assuming $\boldsymbol{x}_v < \boldsymbol{x}_u$, is given by the rule

$$f_{v\to u}^{(t)} = \boldsymbol{x}_u^{(t)}\boldsymbol{x}_v^{(t)} \cdot F_{uv}(\boldsymbol{x}_u^{(t)} - \boldsymbol{x}_v^{(t)}),$$

where $F_{uv}(z) = F_{vu}(z)$ is a function that captures the level of influence between u, v. We assume that $F_{uv} : [-1, 1] \to [-1, 1]$ is continuously differentiable, $F_{uv}(0) = 0$ (there is no population flow between two neighboring types if they have the same fraction of population), is increasing and finally it is odd, i.e., $F_{uv}(-z) = -F_{uv}(z)$ (so that $f_{v\to u}^{(t)} = -f_{u\to v}^{(t)}$).

In this simplified setting we have made the assumption that the system is closed, i.e., that it does not get influence from outside factors (e.g., economical crisis, immigrations flows, terrorism etc).

1.1 Our results and techniques

We first study our migration dynamics without birth and death and settle the open problem of Kempe et al., as it applies to our dynamics.

We show that the dynamics converges set-wise to a fixed point, i.e., there is a set S containing only fixed points such that the distance between the trajectory of the dynamics and S goes to zero for all starting population vectors. To show this convergence result, we use a simple potential function of the population mass namely, the ℓ_2^2 norm of the population vector, and we show that this potential is strictly increasing at each time step (unless the dynamics is at a fixed point). Moreover, the potential is bounded, hence the result follows.

Next, we strengthen this result by showing point-wise convergence as well. The latter result is technically deeper and more difficult, since it means that every trajectory converges to a specific fixed point $\boldsymbol{p}$. We show point-wise convergence by constructing a *local* potential function that is decreasing in a small neighborhood of the limit point $\boldsymbol{p}$. The potential function is always non-zero in that small neighborhood and is zero only at $\boldsymbol{p}$.

Using the latter result and one of the most important theorems in dynamical systems, the Center Stable Manifold Theorem, we prove that with probability one, under an initial population vector picked uniformly at random from the unit simplex, our dynamics converges point-wise to a fixed point $\boldsymbol{p}$, where the *active* types w in $\boldsymbol{p}$, i.e., $w \in V(G)$ so that $\boldsymbol{p}_w > 0$, form an independent set of G. This involves characterization of the linearly stable (see

Section 2.3 for definition) fixed points and proving that the update rule of the dynamics is a local diffeomorphism[2]. This settles the open problem of Kempe et al., mentioned in the Introduction, for our dynamics. This result is important because it allows us to perform a long-term average case analysis of the behavior of our dynamics and make predictions.

Next, we introduce birth and death in our model. Clearly there will be no convergence in this case since new parties are created all the time. Instead we define and study a notion of "stability" which is different from the classical notions that appear in dynamical systems (see Section 2.3 for the definition of the classical notion and Definition 12 for our notion). A dynamics is (T, d)-stable if and only if $\forall t: \ T \leq t \leq T + d$, no population mass moves at step t. We show that despite birth and death, there are arbitrarily long periods of "stability" with high probability, for a sufficiently small p. Finally, we show that in the long run, with high probability, for a sufficiently large p, the number of types in the population will be $O(\log(1/\epsilon))$. This may seem counter-intuitive, since with a large p new types will be created often; however, since new types absorb mass from old types, the old types die frequently. In contrast, in the short term (from the definition of ϵ) we can have up to $\Theta(1/\epsilon)$ types.

Let us give an interpretation of the results of the previous paragraph in terms of political parties of certain countries (information obtained from Wikipedia). Countries do have periods of political stability, e.g., during 1981-85, 2004-07, no new major (with more than 1% of the vote) parties were formed in Greece, moreover there was no substantial change in the percentage of votes won by parties in successive elections. The parameter ϵ can be interpreted as the fraction of people that can form a party that participates in elections. The minimum size of a party arises for organizational and legal reasons, and is $\Theta(1/Q)$, where Q is the population of the country and therefore ϵ is inversely related to population. The message of the latter theorem is that the number of political parties grows at most as the logarithm of the population of the country, i.e. $O(\log Q)$ [3]. The following data supports this fact. The population of Greece, Spain and India in 2015 was $1.1e7, 4.6e7$ and $1.2e9$, respectively, and the number of parties that participated in the general elections was 20, 32 and 50, respectively.

1.2 Related work

As stated above, we build on the work of Kempe et al. [15]. They model their dynamics in a similar way, i.e., there is a flow of population for every interacting pair of types u, v. The flow goes from smaller to bigger types; in our case the mass is just the population of a type. One very interesting common trait between the two dynamics is that the fixed points have similar description: all types with positive mass belonging to the same connected component C have the same mass. Stable fixed points also have the same properties in both dynamics, namely they are independent sets. The update rules of the two dynamics are somewhat different; our simpler dynamics helps us in proving stronger results.

One of the most studied models is the following: there is a graph G in which each vertex denotes an individual having two possible opinions. At each time step, an individual is chosen at random who next chooses his opinion according to the majority (best response) opinion among his neighbors. This has been introduced by Galam[10] and appeared in [22, 9], where they address the question: in which classes of graphs do individuals reach consensus. The same dynamics, but with each agent choosing his opinion according to noisy

[2] Continuously differentiable, the inverse exists and is also continuously differentiable (in some small neighborhood of each point).

[3] This is just an upper bound, countries like UK, US satisfy this rule too.

best response (the dynamics is a Markov chain) has been studied in [21, 16] and many other papers referenced therein. They give bounds for the hitting time and expected time of the consensus state (risk dominant) respectively.

Another well-known model for the dynamics of opinion formation in multi-agent systems is Hegselmann-Krause [11]. Individuals are discrete entities and are modeled as points in some opinion space (e.g., real line). At every time step, each individual moves to the mass center of all the individuals within unit distance. Typical questions are related to the rate of convergence (see [6] and references therein). Finally, another classic model is the voter model, where there is a fixed graph G among the individuals, and at every time step, a random individual selects a random neighbor and adopts his opinion [12]. For more information on opinion formation dynamics of an individual using information learned from his neighbors, see [13] for a survey (also see [29] for more information on opinion formation models).

Other works, including dynamical systems that show convergence to fixed points, are [23, 19, 18, 24, 20, 27]. [27] focuses on quadratic dynamics and they show convergence in the limit. On the other hand [3] shows that sampling from the distribution this dynamics induces at a given time step is PSPACE-complete. In [23, 19], it is shown that replicator dynamics in linear congestion and 2-player coordination games converges to pure Nash equilibria, and in [18, 24] it is shown that gradient descent converges to local minima, avoiding saddle points even in the case where the fixed points are uncountably many.

Organization: In Section 2 we describe our dynamics formally and give the necessary definitions about dynamical systems. In Section 3 we show that our dynamics without births/deaths converges with probability one to fixed points $\boldsymbol{p}$ so that the set of types with positive population, i.e., active types, form an independent set of G. Finally, in Section 4 we first show that there is no explosion in the number of types (i.e., the order never becomes $\Theta(1/\epsilon)$) and also we perform stability analysis using our notion.

2 Preliminaries

Notation: We denote the probability simplex on a set of size n as Δ_n. Vectors in $\mathbb{R}^n$ are denoted in boldface and $\boldsymbol{z}_j$ denotes the jth coordinate of a given vector $\boldsymbol{z}$. Time indices are denoted by superscripts. Thus, a time indexed vector $\boldsymbol{z}$ at time t is denoted as $\boldsymbol{z}^{(t)}$. We use the letters $J, \mathbb{J}$ to denote the Jacobian of a function and finally we use f^t to denote the composition of f by itself t times.

2.1 Migration dynamics

Let $G = (V, E)$ be an undirected graph on n vertices (which we also call types), and let N_v denote the set of neighbors of v in G. During the whole dynamical process, each vertex v has a non-negative population mass representing the fraction of the population of type v. We consider a discrete-time process and let $\boldsymbol{x}_v^{(t)}$ denote the mass of v at time step t. It follows that the condition

$$\sum_{v \in V(G)} \boldsymbol{x}_v^{(t)} = 1,$$

must be maintained for all t, i.e., $\boldsymbol{x}^{(t)} \in \Delta_n$[4] for all $t \in \mathbb{N}$.

[4] Recall that Δ_n denotes the unit simplex of dimension $n+1$, where $|V(G)| = n$.

Additionally, we consider a dynamical migration rule where the population can move along edges of G at each step. The movement at step t is determined by $\boldsymbol{x}^{(t)}$. Specifically, for $uv \in E(G)$, the amount of mass moving from v to u at step t is given by

$$f_{v\to u}^{(t)} = \boldsymbol{x}_u^{(t)} \boldsymbol{x}_v^{(t)} F_{uv}(\boldsymbol{x}_u^{(t)} - \boldsymbol{x}_v^{(t)}).$$

For all $uv \in E(G)$ we assume that $F_{uv} : [-1,1] \to [-1,1]$ is a continuously differentiable function such that:

1. $F_{uv}(0) = 0$ (there is no population flow between two neighboring types if they have the same fraction of population),
2. F_{uv} is increasing (the larger $\boldsymbol{x}_u - \boldsymbol{x}_v$, the more population moving from v to u),
3. F_{uv} is odd i.e., $F_{uv}(-z) = -F_{uv}(z)$ (so that $f_{v\to u}^{(t)} = -f_{u\to v}^{(t)}$).

It can be easily derived from the assumptions that $F_{uv}(z) \geq F_{uv}(0) = 0$ for $z \geq 0$ and $F'_{uv}(-z) = F'_{uv}(z)$ for $z \in [-1,1]$, where F'_{uv} denotes the derivative of F_{uv}. Note that $f_{v\to u}^{(t)} > 0$ implies that population is moving from v to u, and $f_{v\to u}^{(t)} < 0$ implies that population is moving in the other direction. The update rule for the population of type u can be written as

$$\begin{aligned} \boldsymbol{x}_u^{(t+1)} &= \boldsymbol{x}_u^{(t)} + \sum_{v\in N_u} f_{v\to u}^{(t)} && (1)\\ &= \boldsymbol{x}_u^{(t)} + \sum_{v\in N_u} \boldsymbol{x}_u^{(t)} \boldsymbol{x}_v^{(t)} F_{uv}(\boldsymbol{x}_u^{(t)} - \boldsymbol{x}_v^{(t)}). && (2) \end{aligned}$$

We denote the update rule of the dynamics as $g : \Delta_n \to \Delta_n$, i.e., we have that

$$\boldsymbol{x}^{(t+1)} = g(\boldsymbol{x}^{(t)}).$$

Therefore it holds that $\boldsymbol{x}^{(t)} = g^t(\boldsymbol{x}^{(0)})$, where g^t denotes the composition of g by itself t times. It is easy to see g is well-defined for $\sup_{z\in[-1,1]} |F_{uv}(z)| \leq 1$ for all $uv \in E(G)$, in the sense that if $\boldsymbol{x}^{(t)} \in \Delta_n$ then $\boldsymbol{x}^{(t+1)} \in \Delta_n$. This is true since for all u we get (using induction, i.e., $\boldsymbol{x}^{(t)} \in \Delta_n$)

$$\begin{aligned} \boldsymbol{x}_u^{(t+1)} &= \boldsymbol{x}_u^{(t)} + \sum_{v\in N_u} \boldsymbol{x}_u^{(t)} \boldsymbol{x}_v^{(t)} F_{uv}(\boldsymbol{x}_u^{(t)} - \boldsymbol{x}_v^{(t)})\\ &\geq \boldsymbol{x}_u^{(t)} - \sum_{v\in N_u} \boldsymbol{x}_u^{(t)} \boldsymbol{x}_v^{(t)} \geq \boldsymbol{x}_u^{(t)} - \boldsymbol{x}_u^{(t)}(1 - \boldsymbol{x}_u^{(t)}) \geq 0, \end{aligned}$$

moreover it holds

$$\begin{aligned} \boldsymbol{x}_u^{(t+1)} &= \boldsymbol{x}_u^{(t)} + \sum_{v\in N_u} \boldsymbol{x}_u^{(t)} \boldsymbol{x}_v^{(t)} F_{uv}(\boldsymbol{x}_u^{(t)} - \boldsymbol{x}_v^{(t)})\\ &\leq \boldsymbol{x}_u^{(t)} + \sum_{v\in N_u} \boldsymbol{x}_u^{(t)} \boldsymbol{x}_v^{(t)} \leq \boldsymbol{x}_u^{(t)} + \boldsymbol{x}_u^{(t)}(1 - \boldsymbol{x}_u^{(t)}) \leq \boldsymbol{x}_u^{(t)} + 1 - \boldsymbol{x}_u^{(t)} = 1, \end{aligned}$$

and also $\sum_u \boldsymbol{x}_u^{(t+1)} = \sum_u \boldsymbol{x}_u^{(t)} = 1$ (the other terms cancel out).

2.2 Birth and death of types

Political parties or social communities don't tend to survive once their size becomes "small" and hence there is a need to incorporate death of parties in our model. We will define a global parameter ϵ in our model. When the population mass of a type v becomes smaller than some fixed value ϵ, we consider it to be dead and move its mass arbitrarily to existing

types. Formally, if $\boldsymbol{x}_v^{(t)} \leq \epsilon$ then $\boldsymbol{x}_v^{(t)} \leftarrow 0$ and $\boldsymbol{x}_u^{(t)} \leftarrow \boldsymbol{x}_u^{(t)} + \boldsymbol{x}_v^{(t)} / |N_v|$ for all $u \in N_v$. Also, vertex v is removed and edges are added arbitrarily on its neighbors to ensure connectivity of the resulting graph.

▶ Remark. It is not hard to see that the maximum number of types is $1/\epsilon$ (by definition). We say that we have explosion in the number of types if they are of $\Theta(1/\epsilon)$. In Theorem 16 we show that in the long run, the number of types is much smaller – it is $O(\log(1/\epsilon))$ with high probability.

Every so often, new political opinions emerge and like-minded people move from the existing parties to create a new party, which then follows the normal dynamics to either survive or die out. To model birth of new types, at each time step, with probability p, we create a new type v such that v takes a portion of mass from each existing type independently. The amount of mass going to v from each u follows an arbitrary distribution in the range $[\beta_{\min}, \beta_{\max}]$. Specifically, let $\boldsymbol{Z}_u \sim \mathcal{D}$ where $\mathcal{D}$ is a distribution with support $[\beta_{\min}, \beta_{\max}]$, the amount of mass going from u to v is $\boldsymbol{Z}_u \boldsymbol{x}_u$. We connect v to the existing graph arbitrarily such that it remains connected.

Additionally, we make a small change to the migration dynamics defined in Section 2.1 to make it more realistic. Our tenet is that population mass migrates from smaller to bigger types because of influence. However, if the two types are of approximately the same size, the difference is size is not discernible and hence migration should not happen. To incorporate this, we introduce a new parameter $\delta > 0$ and if $|\boldsymbol{x}_u - \boldsymbol{x}_v| \leq \delta$, we assume that no population moves from u to v.

Finally, each step of the dynamics consists of there phases in the following order:

1. Migration: the dynamics follows the update rule from Section 2.1.
2. Birth: with probability p, a new type v is created and takes mass from the existing types.
3. Death: a type with mass smaller than ϵ dies out and move its mass to the existing types.

▶ Remark. For any different order of phases, all proofs in the paper still go through with minimal changes.

2.3 Definitions and basics

A recurrence relation of the form $\boldsymbol{z}^{(t+1)} = f(\boldsymbol{z}^{(t)})$ is a discrete time dynamical system, with update rule $f : \mathcal{S} \to \mathcal{S}$ (for our purposes, the set $\mathcal{S}$ is Δ_n). The point $\boldsymbol{p}$ is called a *fixed point* or *equilibrium* of f if $f(\boldsymbol{p}) = \boldsymbol{p}$. A fixed point $\boldsymbol{p}$ is called *Lyapunov stable* (or just stable) if for every $\varepsilon > 0$, there exists a $\zeta = \zeta(\varepsilon) > 0$ such that for all $\boldsymbol{z}$ with $\|\boldsymbol{z} - \boldsymbol{p}\| < \zeta$ we have that $\left\|f^k(\boldsymbol{z}) - \boldsymbol{p}\right\| < \varepsilon$ for every $k \geq 0$. We call a fixed point $\boldsymbol{p}$ *linearly stable* if, for the Jacobian $J(\boldsymbol{p})$ of f, it holds that its spectral radius is at most one. It is true that if a fixed point $\boldsymbol{p}$ is stable then it is linearly stable but the converse does not hold in general [25]. A sequence $(f^t(\boldsymbol{z}^{(0)}))_{t\in\mathbb{N}}$ is called a *trajectory* of the dynamics with $\boldsymbol{z}^{(0)}$ as starting point. A common technique to show that a dynamical system converges to a fixed point is to construct a function $P : \Delta_m \to \mathbb{R}$ such that $P(f(\boldsymbol{z})) > P(\boldsymbol{z})$ unless $\boldsymbol{z}$ is a fixed point. We call P a *potential* or *Lyapunov* function.

3 Convergence to independent sets almost surely

In this section we prove that the deterministic dynamics (assuming no death/birth of types, namely the graph G remains fixed) converges point-wise to fixed points $\boldsymbol{p}$ where $\{v : \boldsymbol{p}_v > 0\}$ (set of active types) is an independent set of the graph G, with probability one assuming that the starting point $\boldsymbol{x}^{(0)}$ follows an atomless distribution with support in Δ_n. To do that,

we show that for all starting points $\boldsymbol{x}^{(0)}$, the dynamics converges point-wise to fixed points. Moreover we prove that the update rule of the dynamics is a diffeomorphism and that the linearly stable fixed points $\boldsymbol{p}$ of the dynamics satisfy the fact that the set of active types in $\boldsymbol{p}$ is an independent set of G. Finally, our main claim of the section follows by using Center-Stable Manifold theorem.

Structure of fixed points. The fixed points of the dynamics (1) are vectors $\boldsymbol{p}$ such that for each $uv \in E(G)$, at least one of the following conditions must hold:

1. $\boldsymbol{p}_v = \boldsymbol{p}_u$, 2. $\boldsymbol{p}_v = 0$, 3. $\boldsymbol{p}_u = 0$.

Therefore, for each fixed point $\boldsymbol{p}$, the set of active types (types with non-zero population mass) with respect to $\boldsymbol{p}$ must form a set of connected components such that all types in each component have the same population mass. We first prove that the dynamics converges point-wise to fixed points.

3.1 Point-wise convergence

Initially, we consider the function $\Phi(\boldsymbol{x}) = \sum_v \boldsymbol{x}_v^2$ and state the following lemma on Φ.

▶ **Lemma 1 (Lyapunov (potential) function).** *Let $\boldsymbol{x}$ be a point with $\boldsymbol{x}_u > \boldsymbol{x}_v$. Let $\boldsymbol{y}$ be another point such that $\boldsymbol{y}_v = \boldsymbol{x}_v - d$, $\boldsymbol{y}_u = \boldsymbol{x}_u + d$ for some $0 < d \leq \boldsymbol{x}_v$ and $\boldsymbol{y}_z = \boldsymbol{x}_z$ for all $z \neq u, v$. Then $\Phi(\boldsymbol{x}) < \Phi(\boldsymbol{y})$.*

If we think of $\boldsymbol{x}$ as a population vector, Lemma 1 implies that $\Phi(\boldsymbol{x})$ increases if population is moving from a smaller type to a bigger type.

▶ **Theorem 2 (Set-wise convergence).** *$\Phi(\boldsymbol{x}^{(t)})$ is strictly increasing along every nontrivial trajectory, i.e., $\Phi(\boldsymbol{x}^{t+1}) = \Phi(g(\boldsymbol{x}^{(t)})) \geq \Phi(\boldsymbol{x}^{(t)})$ with equality only when $\boldsymbol{x}^{(t)}$ is a fixed point. As a corollary, the dynamics converges to fixed points (set-wise convergence).*

Using the above theorem (Theorem 2) along with the construction of a local Lyapunov function, we can show the following theorem:

▶ **Theorem 3 (Point-wise convergence).** *The dynamics converges point-wise to fixed points.*

Proof Sketch of Theorems 2 and 3. We show Theorem 2 by first breaking the migration step from $\boldsymbol{x}^{(t)}$ to $\boldsymbol{x}^{(t+1)}$ into multiple steps which involve migration between two types only and using Lemma 1. Moreover, given a limit point $\boldsymbol{p}$ of a trajectory with initial population vector $\boldsymbol{x}^{(0)}$, we create a local Lyapunov function Ψ that depends on $\boldsymbol{p}$, i.e., $\Psi(\boldsymbol{x}, \boldsymbol{p}) = \sum_{v:\boldsymbol{p}_v>0}(\boldsymbol{p}_v - \boldsymbol{x}_v)$. Ψ is decreasing and nonnegative in a neighborhood of $\boldsymbol{p}$, and zero only at $\boldsymbol{p}$. Since $\boldsymbol{p}$ is a limit point, there is a subsequence of times $t_k \to \infty$ so that the dynamics for these times converges to $\boldsymbol{p}$, therefore the dynamics converges to $\boldsymbol{p}$ as $t \to \infty$, with initial condition $\boldsymbol{x}^{(0)}$. ◀

3.2 Diffeomorphism and stability analysis via Jacobian

In this section we compute the Jacobian J of g and then perform spectral analysis on J. The Jacobian of g is the following:

$$\frac{\partial g_u}{\partial \boldsymbol{x}_u} = J_{u,u} = 1 + \sum_{v \in N_u} \boldsymbol{x}_v \left[F_{uv}(\boldsymbol{x}_u - \boldsymbol{x}_v) + \boldsymbol{x}_u F'_{uv}(\boldsymbol{x}_u - \boldsymbol{x}_v)\right],$$

$$\frac{\partial g_u}{\partial \boldsymbol{x}_v} = J_{u,v} = \boldsymbol{x}_u \left[F_{uv}(\boldsymbol{x}_u - \boldsymbol{x}_v) - \boldsymbol{x}_v F'_{uv}(\boldsymbol{x}_u - \boldsymbol{x}_v)\right] \text{ if } uv \in E(G) \text{ else } 0.$$

▶ **Lemma 4** (Local Diffeomorphism). *The Jacobian is invertible on the subspace* $\sum_v \boldsymbol{x}_v = 1$, *for* $\sup_{z\in[-1,1]} |F_{uv}(z)| < \frac{1}{2}$ *for each* $uv \in E(G)$. *Moreover,* g *is a local diffeomorphism in a neighborhood of* Δ_n.

▶ **Lemma 5** (Linearly stable fixed point ⇒ independent set). *Let* $\boldsymbol{p}$ *be a fixed point such that there exists a connected component* C *of size greater than 1, and all types* $v \in C$ *have the same positive mass* $\boldsymbol{p}_v > 0$. *Then the Jacobian at* $\boldsymbol{p}$ *has an eigenvalue with absolute value greater than one.*

Proof Sketch of Lemmas 4 and 5. To prove Lemma 4, it suffices to show that the Jacobian $J(\boldsymbol{x})$ is invertible and then use the Inverse Function theorem. Invertibility comes from the fact that $J(\boldsymbol{x})$ is shown to be strictly diagonally dominant for $|F_{uv}(z)| < \frac{1}{2}$ for each $uv \in E(G)$ and $z \in [-1, 1]$. Moreover, to show Lemma 5, we can show that if a fixed point $\boldsymbol{p}$ does not induce an independent set, then the trace of the Jacobian of size $l \times l$ (after removing columns and rows of non-active types) at $\boldsymbol{p}$ is greater than l. Since the trace of a matrix is equal to the sum of its eigenvalues, the maximum eigenvalue in absolute value is greater than one and the claim follows. ◀

▶ Remark. Lemmas 4 and 5 are the key Lemmas for the next subsection in which we prove our first main result (Theorem 6 and Corollary 10). If one wants to prove such a result for the model of Kempe et al., these are the two lemmas that need to be adapted to their setting. Analyzing the Jacobian of the update rule of that model is very challenging since the update rule is a rational function, compared to our model which is generic but the derivatives are simpler to compute and analyze.

3.3 Center-stable manifold and average case analysis

In this section we prove our first main result, Corollary 10, which is a consequence of the following theorem:

▶ **Theorem 6.** *Assume that* $\max_{z\in[-1,1]} |F_{uv}(z)| < 1/2$ *for all* $uv \in E(G)$. *The set of points* $\boldsymbol{x} \in \Delta_n$ *such that dynamics 1 starting at* $\boldsymbol{x}$ *converges to a fixed point* $\boldsymbol{p}$ *whose active types do not form an independent set of* G *has measure zero.*

To prove Theorem 6, we are going to use arguably one of the most important theorems in dynamical systems, called *Center Stable Manifold Theorem*:

▶ **Theorem 7** (Center-stable Manifold Theorem [28]). *Let* $\boldsymbol{p}$ *be a fixed point for the* C^r *local diffeomorphism* $f : U \to \mathbb{R}^m$ *where* $U \subset \mathbb{R}^m$ *is an open neighborhood of* $\boldsymbol{p}$ *in* $\mathbb{R}^m$ *and* $r \geq 1$. *Let* $E^s \oplus E^c \oplus E^u$ *be the invariant splitting of* $\mathbb{R}^m$ *into generalized eigenspaces of the Jacobian* $J(\boldsymbol{p})$ *that correspond to eigenvalues of absolute value less than one, equal to one, and greater than one. To the* $J(\boldsymbol{p})$ *invariant subspace* $E^s \oplus E^c$ *there is an associated local* f *invariant* C^r *embedded disc* W^{sc}_{loc} *tangent to the linear subspace at* $\boldsymbol{p}$ *and a ball* B *around* $\boldsymbol{p}$ *such that:*

$$f(W^{sc}_{loc}) \cap B \subset W^{sc}_{loc}. \text{ If } f^m(\boldsymbol{x}) \in B \text{ for all } m \geq 0, \text{ then } \boldsymbol{x} \in W^{sc}_{loc}. \tag{3}$$

Since an n-dimensional simplex Δ_n in $\mathbb{R}^n$ has dimension $n-1$, we need to take a projection of the domain space ($\sum_v \boldsymbol{x}_v = 1$) and accordingly redefine the map g. Let $\boldsymbol{x}$ be a point mass in Δ_n. Let u be a fixed type and define $h : \mathbb{R}^n \to \mathbb{R}^{n-1}$ so that we exclude the variable $\boldsymbol{x}_u$ from $\boldsymbol{x}$, i.e., $h(\boldsymbol{x}) = \boldsymbol{x}_{-u}$. We substitute the variable $\boldsymbol{x}_u$ with $1 - \sum_{v\neq u} \boldsymbol{x}_v$ and let g' be the resulting update rule of the dynamics $g'(\boldsymbol{x}_{-u}) = g(\boldsymbol{x})$. The following lemma gives a relation between the eigenvalues of the Jacobians of functions g and g'.

▶ **Lemma 8.** *Let J, J' be the Jacobian of g, g' respectively. Let λ be an eigenvalue of J such that λ does not correspond to left eigenvector $(1, \ldots, 1)$ (with eigenvalue 1). Then J' has also λ as an eigenvalue.*

Before we proceed with the proof sketch of Theorem 6 and Corollary 10, we state the following which is a corollary of Lemmas 4, 5 and 8 and also uses classic properties for determinants of matrices.

▶ **Corollary 9.** *Let $\boldsymbol{p}$ be a fixed point whose active types do not form an independent set in G. Then J' at $h(\boldsymbol{p})$ has an eigenvalue with absolute value greater than one. Additionally, the Jacobian J' of g' is invertible in $h(\Delta_n)$ and as a result g' is a local diffeomorphism in a neighborhood of $h(\Delta_n)$.*

▶ **Corollary 10** (Convergence to independent sets)**.** *Suppose that $\max_{z\in[-1,1]} |F_{uv}(z)| < 1/2$ for all $uv \in E(G)$. If the initial mass vector $\boldsymbol{x}^{(0)} \in \Delta_n$ is chosen from an atomless distribution, then the dynamics converges point-wise with probability 1 to a point $\boldsymbol{p}$ whose active types form an independent set in G.*

Proof Sketch of Theorem 6 and Corollary 10. The proof of Corollary 10 comes from Theorem 3 and Theorem 6.

The main steps for the proof of Theorem 6 are as follows: Due to Center-Stable Manifold theorem (we can use it since the update rule of the dynamics is a local diffeomorphism, by Lemma 4 and Lemma 9) we have that the set of initial population vectors that stay trapped in a small enough neighborhood of an unstable fixed point is a lower dimensional manifold, hence a zero measure set. Any initial condition that converges point-wise to this unstable fixed point must at some time t reach points in this set. All of these initial conditions can thus be covered by a countable union of pre-images of the zero measure neighborhood implied by the Center-Stable Manifold theorem. Because the update rule is a local diffeomorphism, these pre-images must also be of zero measure and the countable union of zero measure sets imply a zero measure region of attraction for each unstable equilibrium. The only remaining hurdle is to cover the set of linearly unstable fixed points with a countable cover of the small neighborhoods. Finally, by Lemma 5 any fixed point $\boldsymbol{p}$ whose active types do not form an independent set of G is linearly unstable and the claim follows. ◀

Corollary 10 is illustrated in Figure 1 for the case of a 3-path and a triangle. As shown in the figure, if the initial condition is chosen uniformly at random from a point in the simplex, the dynamics converges to an independent set with probability one.

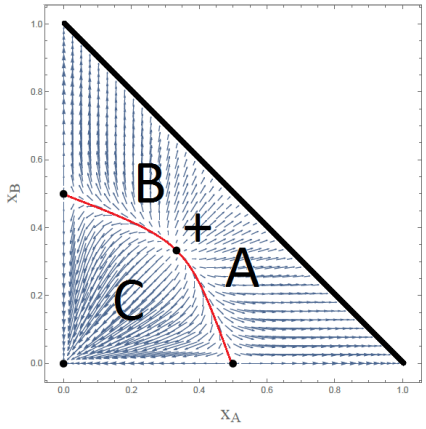
4 Stability and bound on the number of types

In this section we consider dynamical systems with migration, death and birth and prove two probabilistic statements on stability and number of types. The following direct application of Chernoff's bound is used intensively to attain probabilistic guarantees.

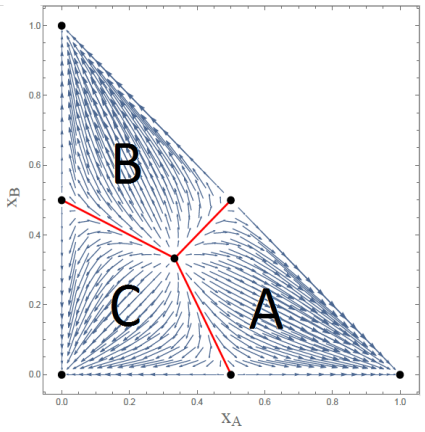
▶ **Lemma 11.** *In a period of t steps, there are at least $tp/2$ births with probability at least $1 - e^{-tp/8}$ and there are at most $3tp/2$ births with probability at least $1 - e^{-tp/6}$.*

4.1 Stability

We define the notion of stability and give a stability result for a dynamical system involving migration, death and birth. For the rest of the paper, we denote by $\alpha_{\min} =$



(a) The region with "C" corresponds to the initial population vectors so that the dynamics converges to the fixed point where all the population is of type C. The region "A+B" corresponds to the initial population masses so that the dynamics converges to a fixed point where part of the population is of type A and the rest of type B.



(b) Each region "A", "B", "C" corresponds to the initial population vectors so that the dynamics converges to all the population being of type A, B, C respectively. It is easy to see that an initial vector $(\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C)$ converges to the fixed point where all population is of type $\arg\max_{i\in\{A,B,C\}} \boldsymbol{x}_i$. In case of ties, the limit population is split equally among the tied types (symmetry).

Figure 1 Migration dynamics phase portrait for path and triangle of 3 types A, B, C respectively and for $F_{uv}(z) = 0.25z$ for all $uv \in E(G)$. The black points and the line correspond to the fixed points. $\boldsymbol{x}_A, \boldsymbol{x}_B$ correspond to the fractions of people that are of type A, B. We omit $\boldsymbol{x}_C$ since $\boldsymbol{x}_C = 1 - \boldsymbol{x}_A - \boldsymbol{x}_B$.

$\min_{uv\in E(G), z\in[-1,1]} F'_{uv}(z)$ and $\alpha_{\max} = \max_{uv\in E(G), z\in[-1,1]} F'_{uv}(z)$. $\alpha_{\min}, \alpha_{\max}$ are non-negative and finite since F_{uv} is continuously differentiable, increasing and $[-1, 1]$ is a compact set. It can be seen easily that for each $uv \in E(G)$ and $z \in [-1, 1]$,

$$\alpha_{\min}(z - 0) \le F_{uv}(z) - F_{uv}(0) \le \alpha_{\max}(z - 0).$$

Since $F_{uv}(0) = 0$, $\alpha_{\min} z \le F_{uv}(z) \le \alpha_{\max} z$.

▶ **Definition 12** ((T, d)-Stable dynamics). A dynamics is (T, d)-stable if and only if $\forall T \le t \le T + d$, no population mass moves in the migration phase at step t.

We state the following two lemmas whose proofs come from the definition of Φ.

▶ **Lemma 13.** *If the dynamics is not $(t, 0)$-stable, the migration phase at time t increases Φ by at least $2\alpha_{\min}\epsilon\delta^3$.*

▶ **Lemma 14.** *Each birth can decrease Φ by at most $2\beta_{\max}$.*

With the two above lemmas, we can give a theorem on the "stability" of the dynamics:

▶ **Theorem 15** ("Stable" for long enough). *Assume $\alpha_{\min} > 0$. Let $p < \min\left(\frac{\epsilon\delta^3\alpha_{\min}}{3\beta_{\max}}, \frac{2}{3}\right)$ and $t > \frac{1}{\epsilon\delta^3\alpha_{\min} - 3p\beta_{\max}}$. With probability at least $1 - e^{-tp/6}$, the dynamics is $\left(T, \frac{1}{3p}\right)$-stable for some $T \le t$.*

Proof Sketch. Consider a period of t steps. Lemma 11 guarantees that there are at most $3tp/2$ births in the period with the desired probability. In the migration phases of the period, Φ can either increase if there is a migration or remain unchanged otherwise.

Assume that Φ increases in more than $t/2$ migration phases. Using Lemma 13 and Lemma 14, we can bound the net increase of Φ in the period. Specifically, the net increase is least $t(\alpha_{\min}\epsilon\delta^3 - 3p\beta_{\max})$, which is greater than 1. Therefore, we have reached a contradiction.

It follows that Φ cannot increase in more than $t/2$ migration phases, and must remain unchanged in at least $t/2$ migration phases. Since there are at most $3tp/2$ births, there must be no migration in a period of $1/(3p)$ consecutive steps. ◀

4.2 Bound on the number of types

In this section we investigate a behavior of the dynamics following a long period of time. Specifically, we show that after a large number of steps, the number of types can not be too high. Our goal is to prove the following theorem:

▶ **Theorem 16** (Lack of explosion). *Let $\alpha_{\max} \leq p/512$ and $t \geq (16/p)\log^2(1/\epsilon)$. The dynamics at step t has at most $72\log(1/\epsilon)$ types with probability at least $1 - 3\epsilon$.*

First we give the following lemma, which says that if the number of types is large enough, then after a fixed period of time, it will decrease by a factor of roughly 2.

▶ **Lemma 17.** *Let $\alpha_{\max} \leq p/512$ and k be the number of types at step t_0. If $k \geq 48\log(1/\epsilon)$, with probability at least $1 - 2\epsilon^2$, the number of types at step $t_0 + (16/p)\log(1/\epsilon)$ is at most $k/2 + 24\log(1/\epsilon)$.*

Proof Sketch of Theorem 16. Consider the last $(16/p)\log^2(1/\epsilon)$ steps of the dynamics. We call a period of $(16/p)\log(1/\epsilon)$ steps a *decreasing period* if it satisfies the condition in Lemma 17, i.e, if the number of types k at the beginning of the period is at least $48\log(1/\epsilon)$, and the number of types at the end of the period is at most $k/2 + 24\log(1/\epsilon)$.

Construct a set P of periods of length $(16/p)\log(1/\epsilon)$ as follows. Start with $t' = 0$ and repeat the following step until $t' = t$. If $t' + (16/p)\log(1/\epsilon) \leq t$ and the number of types at t' is at least $48\log(1/\epsilon)$, let i be the period from t' to $t' + (16/p)\log(1/\epsilon)$, and add i to P. Update $t' \leftarrow t' + (16/p)\log(1/\epsilon)$. Else update $t' \leftarrow t' + 1$.

Assume that all periods in P are decreasing periods. By Lemma 17, the probability of such an outcome occurring is at least $1 - 2\epsilon$. With that assumption, if the number of types ever becomes smaller than $48\log(1/\epsilon)$ and reaches $48\log(1/\epsilon)$ again, it will be at least $48\log(1/\epsilon)$ after a period of $(16/p)\log(1/\epsilon)$ steps unless there are less than $(16/p)\log(1/\epsilon)$ subsequent steps. In that case, by Lemma 11, the probability that in the remaining steps, there are at most $24\log(1/\epsilon)$ births is at least $1 - \epsilon$. By union bound, the probability of both outcomes occurring is at least $1 - 3\epsilon$.

Moreover, since the number of types at the beginning is at most $1/\epsilon$, with the assumption, it must become smaller than $48\log(1/\epsilon)$ at some step of the dynamics. The theorem then follows. ◀

References

1 A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 7–15, 2008.

2 S. Aral, L. Muchnik, and A. Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2009.

3 S. Arora, Y. Rabani, and U. Vazirani. Simulating quadratic dynamical systems is pspace-complete (preliminary version). *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 459–467, 1994.

4 R. Axelrod. The dissemination of culture. *Journal of Conflict Resolution*, pages 203–226, 1997.

5 E. Bakshy, I. Rosenn, C. A. Marlow, and L. A. Adamic. The role of social networks in information diffusion. *21st International World Wide Web Conference*, pages 203–226, 2012.

6 A. Bhattacharyya and K. Shiragur. How friends and non-determinism affect opinion dynamics. In *54th IEEE Conference on Decision and Control (CDC)*, pages 6466–6471, 2015.

7 J. M. Cohen. Sources of peer group homogeneity. *Sociology in Education*, pages 227–241, 1977.

8 G. Deffuant, D. Neau, F. Amblard, and G. Weisbuch. Mixing beliefs among interacting agents. *Journal of Conflict Resolution*, pages 87–98, 2000.

9 M. Feldman, N. Immorlica, B. Lucier, and S. M. Weinberg. Reaching consensus via non-bayesian asynchronous learning in social networks. In *APPROX/RANDOM*, pages 192–208, 2014.

10 S. Galam. Sociophysics: a review of Galam models. *International Journal of Modern Physics C*, 2008.

11 R. Hegselmann and U. Krause. Opinion dynamics and bounded confidence: models, analysis and simulation. *Journal of Artificial Societies and Social Simulation*, 2002.

12 R. A. Holley and T. M. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *Annals of Probability*, 1975.

13 M. O. Jackson. *Social and economic networks*. Princeton University Press, 2008.

14 D. B. Kandel. Homophily, selection, and socialization in adolescent friendships. *American Journal of Sociology*, pages 427–436, 1978.

15 D. Kempe, J. M. Kleinberg, S. Oren, and A. Slivkins. Selection and influence in cultural dynamics. In *ACM Conference on Electronic Commerce (EC)*, pages 585–586, 2013.

16 G. E. Kreindlera and H. P. Young. The spread of innovations in social networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2014.

17 T. LaFond and J. Neville. Randomization tests for distinguishing social influence and homophily effects. *19th International World Wide Web Conference*, pages 601–610, 2010.

18 J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent only converges to minimizers. *Conference on Learning Theory (COLT)*, 2016.

19 R. Mehta, I. Panageas, and G. Piliouras. Natural selection as an inhibitor of genetic diversity: Multiplicative weights updates algorithm and a conjecture of haploid genetics. *Innovations in Theoretical Computer Science (ITCS)*, 2015.

20 R. Mehta, I. Panageas, G. Piliouras, P. Tetali, and V. V. Vazirani. Mutation, sexual reproduction and survival in dynamic environments. *Innovations in Theoretical Computer Science (ITCS)*, 2017.

21 A. Montanari and A. Saberi. The spread of innovations in social networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2010.

22 E. Mossel, J. Neeman, and O. Tamuz. Majority dynamics and aggregation of information in social networks. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

23 I. Panageas and G. Piliouras. Average case performance of replicator dynamics in potential games via computing regions of attraction. *ACM Conference on Economics and Computation (EC)*, 2016.

24 I. Panageas and G. Piliouras. Gradient descent only converges to minimizers: Non-isolated critical points and invariant regions. *Innovations in Theoretical Computer Science (ITCS)*, 2017.

25 L. Perko. *Differential Equations and Dynamical Systems*. Springer, 3nd. edition, 1991.

26 K. Poole and H. Rosenthall. Patterns of congressional voting. *American Journal of Political Science*, pages 228–278, 1978.

27 Y. Rabinovich, A. Sinclair, and A. Widgerson. Quadratic dynamical systems. *Proc 23rd IEEE Symp Foundations of Computer Science*, pages 304–313, 1992.

28 M. Shub. *Global Stability of Dynamical Systems*. Springer-Verlag, 1987.

29 A. Sîrbu, V. Loreto, V. Domenico P. Servedio, and F. Tria. Opinion dynamics: models, extensions and external effects. *CoRR*, abs/1605.06326, 2016.

Hardness of Computing and Approximating Predicates and Functions with Leaderless Population Protocols*

Amanda Belleville[1], David Doty[2], and David Soloveichik[3]

1 **Computer Science, University of California, Davis, CA, USA**
acbelleville@ucdavis.edu

2 **Computer Science, University of California, Davis, CA, USA**
doty@ucdavis.edu

3 **Electrical and Computer Engineering, University of Texas, Austin, TX, USA**
david.soloveichik@utexas.edu

Abstract

Population protocols are a distributed computing model appropriate for describing massive numbers of agents with very limited computational power (finite automata in this paper), such as sensor networks or programmable chemical reaction networks in synthetic biology. A population protocol is said to require a leader if every valid initial configuration contains a single agent in a special "leader" state that helps to coordinate the computation. Although the class of predicates and functions computable with probability 1 (stable computation) is the same whether a leader is required or not (semilinear functions and predicates), it is not known whether a leader is necessary for fast computation. Due to the large number of agents n (synthetic molecular systems routinely have trillions of molecules), efficient population protocols are generally defined as those computing in polylogarithmic in n (parallel) time. We consider population protocols that start in leaderless initial configurations, and the computation is regarded finished when the population protocol reaches a configuration from which a different output is no longer reachable.

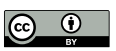
In this setting we show that a wide class of functions and predicates computable by population protocols are not *efficiently* computable (they require at least linear time), nor are some linear functions even efficiently *approximable*. It requires at least linear time for a population protocol even to approximate division by a constant or subtraction (or any linear function with a coefficient outside of $\mathbb{N}$), in the sense that for sufficiently small $\gamma > 0$, the output of a sublinear time protocol can stabilize outside the interval $f(m)(1 \pm \gamma)$ on infinitely many inputs m. In a complementary positive result, we show that with a sufficiently large value of γ, a population protocol *can* approximate any linear f with nonnegative rational coefficients, within approximation factor γ, in $O(\log n)$ time. We also show that it requires linear time to exactly compute a wide range of semilinear functions (e.g., $f(m) = m$ if m is even and $2m$ if m is odd) and predicates (e.g., parity, equality).

1998 ACM Subject Classification C.2.4 Distributed Systems

Keywords and phrases population protocol, time lower bound, stable computation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.141

* The first two authors were supported by NSF grant 1619343 and the third author by NSF grant 1618895.



44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 141; pp. 141:1–141:14

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

1 Introduction

Population protocols were introduced by Angluin, Aspnes, Diamadi, Fischer, and Peralta[3] as a model of distributed computing in which the agents have very little computational power and no control over their schedule of interaction with other agents. They can be thought of as a special case of a model of concurrent processing introduced in the 1960s, known alternately as vector addition systems[16], Petri nets[19], or commutative semi-Thue systems (or, when all transitions are reversible, "commutative semigroups")[9, 17]. As well as being an appropriate model for electronic computing scenarios such as sensor networks, they are a useful abstraction of "fast-mixing" physical systems such as animal populations[22], gene regulatory networks[8], and chemical reactions.

The latter application is especially germane: several recent wet-lab experiments demonstrate the systematic engineering of custom-designed chemical reactions [23, 12, 7, 20], unfortunately in all cases having a cost that scales linearly with the number of unique chemical species (states). (The cost can even be quadratic if certain error-tolerance mechanisms are employed [21].) Thus, it is imperative in implementing a molecular computational system to keep the number of distinct chemical species at a minimum. On the other hand, it is common (and relatively cheap) for the total number of such molecules (agents) to number in the trillions in a single test tube. It is thus important to understand the computational power enabled by a large number of agents n, where each agent has only a constant number of states (each agent is a finite state machine).

A population protocol is said to require a leader if every valid initial configuration contains a single agent in a special "leader" state that helps to coordinate the computation. Studying computation without a leader is important for understanding essentially distributed systems where symmetry breaking is difficult. Further, in the chemical setting obtaining single-molecule precision in the initial configuration is difficult. Thus, it would be highly desirable if the population protocol did not require an exquisitely tuned initial configuration.

1.1 Introduction to the model

A population protocol is defined by a finite set Λ of *states* that each agent may have, together with a *transition function*[1] $\delta : \Lambda^2 \to \Lambda^2$. A *configuration* is a nonzero vector $\mathbf{c} \in \mathbb{N}^\Lambda$ describing, for each $\overline{s} \in \Lambda$, the *count* $\mathbf{c}(\overline{s})$ of how many agents are in state $\overline{s}$. By convention we denote the number of agents by $n = \|\mathbf{c}\| = \sum_{\overline{s}\in\Lambda} \mathbf{c}(\overline{s})$. Given states $\overline{r}_1, \overline{r}_2, \overline{p}_1, \overline{p}_2 \in \Lambda$, if $\delta(\overline{r}_1, \overline{r}_2) = (\overline{p}_1, \overline{p}_2)$ (denoted $\overline{r}_1, \overline{r}_2 \to \overline{p}_1, \overline{p}_2$), and if a pair of agents in respective states $\overline{r}_1$ and $\overline{r}_2$ interact, then their states become $\overline{p}_1$ and $\overline{p}_2$.[2] The next pair of agents to interact is chosen uniformly at random. The expected (parallel) time for any event to occur is the expected number of interactions, divided by the number of agents n. This measure of time is based on the natural parallel model where each agent participates in a constant number of interactions in one unit of time; hence $\Theta(n)$ total interactions are expected per unit time [5].

The most well-studied population protocol task is computing Boolean-valued *predicates*. It is known that a protocol *stably decides* a predicate $\phi : \mathbb{N}^k \to \{0, 1\}$ (meaning computes

[1] Some work allows nondeterministic transitions, in which the transition function maps to subsets of $\Lambda \times \Lambda$. Our results are independent of whether transitions are nondeterministic, and we choose a deterministic, symmetric transition function, rather than a more general relation $\delta \subseteq \Lambda^4$, merely for notational convenience.

[2] In the most generic model, there is no restriction on which agents are permitted to interact. If one prefers to think of the agents as existing on nodes of a graph, then it is the complete graph K_n for a population of n agents.

the correct answer with probability 1; see Section 6 for a formal definition) if [3] and only if [4] ϕ is semilinear.

Population protocols can also compute integer-valued *functions* $f : \mathbb{N}^k \to \mathbb{N}$. Suppose we start with $m \leq n/2$ agents in "input" state $\overline{x}$ and the remaining agents in a "quiescent" state $\overline{q}$. Consider the protocol with a single transition rule $\overline{x}, \overline{q} \to \overline{y}, \overline{y}$. Eventually exactly $2m$ agents are in the "output" state $\overline{y}$, so this protocol computes the function $f(m) = 2m$. Furthermore (letting $\#\overline{s}$ = count of state $\overline{s}$), if $\#\overline{q} - 2m = \Omega(n)$ initially (e.g., $\#\overline{q} = 3m$), then it takes $\Theta(\log n)$ expected time until $\#\overline{y} = 2m$. Similarly, the transition rule $\overline{x}, \overline{x} \to \overline{y}, \overline{q}$ computes the function $f(m) = \lfloor m/2 \rfloor$, but exponentially slower, in expected time $\Theta(n)$. The transitions $\overline{x}_1, \overline{q} \to \overline{y}, \overline{q}$ and $\overline{x}_2, \overline{y} \to \overline{q}, \overline{q}$ compute $f(m_1, m_2) = m_1 - m_2$ (assuming $m_1 \geq m_2$), also in time $\Theta(n)$ if $m_1 = m_2 + O(1)$.

Formally, we say a population protocol *stably computes* a function $f : \mathbb{N}^k \to \mathbb{N}$ if, for every "valid" initial configuration $\mathbf{i} \in \mathbb{N}^\Lambda$ representing input $\mathbf{m} \in \mathbb{N}^k$ (via counts $\mathbf{i}(\overline{x}_1), \ldots, \mathbf{i}(\overline{x}_k)$ of "input" states $\Sigma = \{\overline{x}_1, \ldots, \overline{x}_k\} \subseteq \Lambda$) with probability 1 the system reaches from $\mathbf{i}$ to $\mathbf{o}$ such that $\mathbf{o}(\overline{y}) = f(\mathbf{m})$ ($\overline{y} \in \Lambda$ is the "output" state) and $\mathbf{o}'(\overline{y}) = \mathbf{o}(\overline{y})$ for every $\mathbf{o}'$ reachable from $\mathbf{o}$ (i.e., $\mathbf{o}$ is *stable*). Defining what constitutes a "valid" initial configuration (i.e., what non-input states can be present initially, and how many) is nontrivial. In this paper we focus on population protocols without a *leader*—a state present in count 1, or small count—in the initial configuration. Here, we equate "leaderless" with initial configurations in which no positive state count is sublinear in the population size n.

It is known that a function $f : \mathbb{N}^k \to \mathbb{N}$ is stably computable by a population protocol if and only if its *graph* $\{(\mathbf{m}, f(\mathbf{m})) \mid \mathbf{m} \in \mathbb{N}^k\} \subset \mathbb{N}^{k+1}$ is a semilinear set [4, 11]. This means intuitively that it is piecewise affine, with each affine piece having rational slopes.

Despite the exact characterization of predicates and functions stably computable by population protocols, we still lack a full understanding of which of the stably computable (i.e., semilinear) predicates and functions are computable *quickly* (say, in time polylogarithmic in n) and which are only computable slowly (linear in n). For positive results, significantly more is known about time to *convergence* [5] with a leader (time to reach a configuration with the correct answer). In this paper we shed new light on time to *stabilization* without a leader (time to reach a configuration from which the answer is *guaranteed to remain correct*).

1.2 Contributions

Definition of function computation and approximation. We formally define computation and approximation of functions $f : \mathbb{N}^k \to \mathbb{N}$ for population protocols. This mode of computation was discussed briefly in the first population protocols paper[3, Section 3.4], which focused more on Boolean predicate computation, and it was defined formally in the more general model of chemical reaction networks[11, 13]. Some subtle issues arise that are unique to population protocols. We also formally define a notion of function *approximation* with population protocols, which has its own issues.

Inapproximability of most linear functions with sublinear time and sublinear error. Recall that the transition rule $\overline{x}, \overline{x} \to \overline{y}, \overline{q}$ computes $f(m) = \lfloor m/2 \rfloor$ in linear time. Consider the transitions $\overline{a}, \overline{x} \to \overline{b}, \overline{y}$ and $\overline{b}, \overline{x} \to \overline{a}, \overline{q}$, starting with $\#\overline{x} = m$, $\#\overline{a} = \gamma m$ for some $0 < \gamma < 1$, and $\#\overline{y} = \#\overline{q} = 0$ (so $n = m + \gamma m$ total agents). Then eventually $\#\overline{y} \in \{m/2, \ldots, m/2 + \gamma m\}$ and $\#\overline{x} = 0$ (stabilizing $\#\overline{y}$), after $O(\frac{1}{\gamma} \log n)$ expected time. (This is analyzed in more detail in Section 5.) Thus, if we tolerate an error linear in n, then f can be approximated in logarithmic time. However, Theorem 4.1 shows this error bound to be tight: *any* leaderless population protocol that approximates $f(m) = \lfloor m/2 \rfloor$, or any other linear function with a

coefficient outside of $\mathbb{N}$ (such as $\lfloor 4m/3 \rfloor$ or $m_1 - m_2$), requires at least linear time to achieve sublinear error.

As a corollary, such functions cannot be stably computed in sublinear time (since computing exactly is the same as approximating with zero error). Conversely, it is simple to show that any linear function with all coefficients in $\mathbb{N}$ is stably computable in logarithmic time (Observation 5.1). Thus we have a dichotomy theorem for the efficiency (with regard to stabilization) of computing linear functions f by leaderless population protocols: if all of f's coefficients are in $\mathbb{N}$, then it is computable in logarithmic time, and otherwise it requires linear time.

Approximability of nonnegative rational-coefficient linear functions with logarithmic time and linear error. Theorem 4.1 says that no linear function with a coefficient outside of $\mathbb{N}$ can be stably computed with sublinear time and sublinear error. In a complementary positive result, Theorem 5.2, by relaxing the error to linear, and restricting the coefficients to be *nonnegative* rationals (but not necessarily integers), we show how to approximate any such linear function in logarithmic time. (It is open if $m_1 - m_2$ can be approximated with linear error in logarithmic time.)

Uncomputability of many nonlinear functions in sublinear time. What about non-linear functions? Theorem 3.1 states that sublinear time computation cannot go much beyond linear functions with coefficients in $\mathbb{N}$. We show any function computable in sublinear time is *eventually*-$\mathbb{N}$-*linear*, which we define to be linear with nonnegative integer coefficients on all sufficiently large inputs. Examples of non-eventually-$\mathbb{N}$-linear functions, that provably cannot be computed in sublinear time, include $f(m_1, m_2) = \min(m_1, m_2)$ (computable slowly via $\overline{x}_1, \overline{x}_2 \to \overline{y}, \overline{q}$), and $f(m) = m - 1$ (computable slowly via $\overline{x}, \overline{x} \to \overline{x}, \overline{y}$).

The only remaining semilinear functions whose asymptotic time complexity remains unknown are those "piecewise linear" functions that switch between pieces only near the boundary of $\mathbb{N}^k$; for example, $f(m) = 0$ if $m \leq 3$ and $f(m) = m$ otherwise.

Undecidability of many predicates in sublinear time. Every semilinear predicate $\phi : \mathbb{N}^k \to \{0, 1\}$ is stably decidable in $O(n)$ time [5]. Some, such as $\phi(m) = 1$ iff $m \geq 1$, are stably decidable in $O(\log n)$ time by a leaderless protocol, in this case by the transition $\overline{x}, \overline{q} \to \overline{x}, \overline{x}$, where $\overline{x}$ "votes" for output 1 and $\overline{q}$ votes 0. A predicate is *eventually constant* if $\phi(\mathbf{m}_0) = \phi(\mathbf{m}_1)$ for all sufficiently large $\mathbf{m}_0, \mathbf{m}_1$. We show that if a leaderless population protocol stably decides a predicate ϕ in sublinear time, then ϕ is eventually constant. Examples of non-eventually constant predicates include parity ($\phi(m) = 1$ iff m is odd), majority ($\phi(m_1, m_2) = 1$ iff $m_1 \geq m_2$), and equality ($\phi(m_1, m_2) = 1$ iff $m_1 = m_2$). It does *not* include certain semilinear predicates, such as $\phi(m) = 1$ iff $m \geq 1$ (decidable in $O(\log n)$ time) or $\phi(m) = 1$ iff $m \geq 2$ (decidable in $O(n)$ time, and no faster protocol is known).

Note that there is a fundamental difficulty in extending the last two stated negative results to functions and predicates that "do something different only near the boundary of $\mathbb{N}^k$". This is because for inputs where one state is present in small count, the population protocol could in principle use that input as a "leader state"—and no longer be leaderless.

It is possible that the non-eventually constant predicates and non-eventually-$\mathbb{N}$-linear functions, which cannot be computed in sublinear time in our setting, *could* be efficiently computed in the following ways: (1) *With an initial leader* stabilizing to the correct answer in sublinear time, (2) Without initial leaders but *converging* to the correct output in sublinear time. (3) (With or without a leader) stabilizing to an output in sublinear time but *allowing a small probability of incorrect output.*

1.3 Related work

Positive results. Angluin, Aspnes, Diamadi, Fischer, and Peralta [3] showed that any semilinear predicate can be decided in expected parallel time $O(n \log n)$, later improved to $O(n)$ by Angluin, Aspnes, and Eisenstat [5]. More strikingly, the latter paper showed that if an initial leader is present (a state assigned to only a single agent in every valid initial configuration), then there is a protocol for ϕ that *converges* to the correct answer in expected time $O(\log^5 n)$. However, this protocol's expected time to *stabilize* is still provably $\Omega(n)$. Chen, Doty, and Soloveichik [11] showed in the related model of chemical reaction networks (borrowing techniques from the related predicate results [3, 4]) that any semilinear *function* (integer-output $f : \mathbb{N}^k \to \mathbb{N}$) can similarly be computed with expected convergence time $O(\log^5 n)$ if an initial leader is present, but again with much slower stabilization time $O(n \log n)$. Doty and Hajiaghayi [13] showed that any semilinear function can be computed by a chemical reaction network without a leader with expected convergence and stabilization time $O(n)$. Although the chemical reaction network model is more general, these results hold for population protocols.

Since efficient computation seems to be helped by a leader, the computational task of leader election has received significant recent attention. In particular, Alistarh and Gelashvili [2] showed that in a variant of the model allowing the number of states λ_n to grow with the population size n, a protocol with $\lambda_n = O(\log^3 n)$ states can elect a leader with high probability in $O(\log^3 n)$ expected time. Alistarh, Aspnes, Eisenstat, Gelashvili, and Rivest [1] later showed how to reduce the number of states to $\lambda_n = O(\log^2 n)$, at the cost of increasing the expected time to $O(\log^{5.3} n \log \log n)$.

Negative results. The first attempt to show the limitations of sublinear time population protocols, using the more general model of chemical reaction networks, was made by Chen, Cummings, Doty, and Soloveichik [10]. They studied a variant of the problem in which negative results are easier to prove, an "adversarial worst-case" notion of sublinear time: the protocol is required to be sublinear time not only from the initial configuration, but also from any reachable configuration. They showed that the predicates computable in this manner are precisely those whose output depends only on the presence or absence of states (and not on their exact positive counts). Doty and Soloveichik [14] showed the first $\Omega(n)$ lower bound on expected time from valid initial configurations, proving that any protocol electing a leader with probability 1 takes $\Omega(n)$ time.

These techniques were recently improved by Alistarh, Aspnes, Eisenstat, Gelashvili, and Rivest [1], who showed that even with up to $\lambda_n = O(\log \log n)$ states, any protocol electing a leader with probability 1 requires nearly linear time: $\Omega(n/\text{polylog } n)$. They used these tools to prove time lower bounds for another important computational task: majority (detecting whether state $\overline{x}_1$ or $\overline{x}_2$ is more numerous in the initial population, by stabilizing on a configuration in which the state with the larger initial count occupies the whole population).

In contrast to these previous results on the specific tasks of leader election and majority, we obtain time lower bounds for a broad class of functions and predicates, showing "most" of those computable at all by population protocols, cannot be computed in sublinear time. Since they all *can* be computed in linear time, this settles their asymptotic population protocol time complexity.

Informally, one explanation for our result could be that some computation requires electing "leaders" as part of the computation, and other computation does not. Since leader election itself requires linear time as shown in [14], the computation that requires it is necessarily inefficient. It is not clear, however, how to define the notion of a predicate or function

computation requiring electing a leader somewhere in the computation, but recent work by Michail and Spirakis helps to clarify the picture [18].

2 Preliminaries

If Λ is a finite set (in this paper, of *states*, which will be denoted as lowercase Roman letters with an overbar such as $\overline{s}$), we write $\mathbb{N}^\Lambda$ to denote the set of functions $\mathbf{c} : \Lambda \to \mathbb{N}$. Equivalently, we view an element $\mathbf{c} \in \mathbb{N}^\Lambda$ as a vector of $|\Lambda|$ nonnegative integers, with each coordinate "labeled" by an element of Λ. (By assuming some canonical ordering $\overline{s}_1, \ldots, \overline{s}_k$ of Λ, we also interpret $\mathbf{c} \in \mathbb{N}^\Lambda$ as a vector $\mathbf{c} \in \mathbb{N}^k$.) Given $\overline{s} \in \Lambda$ and $\mathbf{c} \in \mathbb{N}^\Lambda$, we refer to $\mathbf{c}(\overline{s})$ as the *count of* $\overline{s}$ *in* $\mathbf{c}$. Let $\|\mathbf{c}\| = \|\mathbf{c}\|_1 = \sum_{\overline{s}\in\Lambda} \mathbf{c}(\overline{s})$. We write $\mathbf{c} \leq \mathbf{c}'$ to denote that $\mathbf{c}(\overline{s}) \leq \mathbf{c}'(\overline{s})$ for all $\overline{s} \in \Lambda$. Since we view vectors $\mathbf{c} \in \mathbb{N}^\Lambda$ equivalently as multisets of elements from Λ, if $\mathbf{c} \leq \mathbf{c}'$ we say $\mathbf{c}$ is a *subset* of $\mathbf{c}'$. For $\alpha > 0$, we say that $\mathbf{c} \in \mathbb{N}^k$ is α-*dense* if, for all $i \in \{1, \ldots, k\}$, if $\mathbf{c}(i) > 0$, then $\mathbf{c}(i) \geq \alpha\|\mathbf{c}\|$.

It is sometimes convenient to use multiset notation to denote vectors, e.g., $\{\overline{x}, \overline{x}, \overline{y}\}$ and $\{2\overline{x}, \overline{y}\}$ both denote the vector $\mathbf{c}$ defined by $\mathbf{c}(\overline{x}) = 2$, $\mathbf{c}(\overline{y}) = 1$, and $\mathbf{c}(\overline{s}) = 0$ for all $\overline{s} \notin \{\overline{x}, \overline{y}\}$. Given $\mathbf{c}, \mathbf{c}' \in \mathbb{N}^\Lambda$, we define the vector component-wise operations of addition $\mathbf{c} + \mathbf{c}'$, subtraction $\mathbf{c} - \mathbf{c}'$, and scalar multiplication $m\mathbf{c}$ for $m \in \mathbb{N}$. For a set $\Delta \subset \Lambda$, we view a vector $\mathbf{c} \in \mathbb{N}^\Delta$ equivalently as a vector $\mathbf{c} \in \mathbb{N}^\Lambda$ by assuming $\mathbf{c}(\overline{s}) = 0$ for all $\overline{s} \in \Lambda \setminus \Delta$. Write $\mathbf{c} \restriction \Delta$ to denote the vector $\mathbf{d} \in \mathbb{N}^\Delta$ such that $\mathbf{c}(\overline{s}) = \mathbf{d}(\overline{s})$ for all $\overline{s} \in \Delta$. In this paper, the floor function $\lfloor\cdot\rfloor : \mathbb{R} \to \mathbb{Z}$ is defined to be the integer *closest to 0* that is distance < 1 from the input, e.g., $\lfloor -3.4 \rfloor = -3$ and $\lfloor 3.4 \rfloor = 3$.

We say a function $f : \mathbb{N}^k \to \mathbb{N}$ is *eventually-*$\mathbb{N}$*-affine* if there are $b, c_1, \ldots, c_k \in \mathbb{N}$ and $m_0 \in \mathbb{N}$ such that for all $\mathbf{m} \in \mathbb{N}^k_{\geq m_0}$, $f(\mathbf{m}) = b + \sum_{i=1}^k c_i \mathbf{m}(i)$. We say a function $f : \mathbb{N}^k \to \mathbb{N}$ is *eventually-*$\mathbb{N}$*-linear* if it is eventually-$\mathbb{N}$-affine with offset $b = 0$, i.e., if $f(\mathbf{0}) = 0$. We say the function is $\mathbb{N}$*-linear* if it is eventually-$\mathbb{N}$-linear with $m_0 = 0$. Similarly, a function is $\mathbb{Q}_{\geq 0}$*-linear* if there are $c_1, \ldots, c_k \in \mathbb{Q}_{\geq 0}$ such that for all $\mathbf{m} \in \mathbb{N}^k$, $f(\mathbf{m}) = \sum_{i=1}^k \lfloor c_i \mathbf{m}(i) \rfloor$.

2.1 Population Protocols

A *population protocol* is a pair $\mathcal{P} = (\Lambda, \delta)$, where Λ is a finite set of *states* and $\delta : \Lambda^2 \to \Lambda^2$ is the (symmetric) *transition function*. A *configuration* of a population protocol is a vector $\mathbf{c} \in \mathbb{N}^\Lambda$, with the interpretation that $\mathbf{c}(\overline{s})$ agents are in state $\overline{s} \in \Lambda$. If there is some "current" configuration $\mathbf{c}$ understood from context, we write $\#\overline{s}$ to denote $\mathbf{c}(\overline{s})$. By convention, the value $n \in \mathbb{Z}_{\geq 1}$ represents the total number of agents $\|\mathbf{c}\|$. A *transition* is a 4-tuple $\tau = (\overline{r}_1, \overline{r}_2, \overline{p}_1, \overline{p}_2) \in \Lambda^4$, written $\tau : \overline{r}_1, \overline{r}_2 \to \overline{p}_1, \overline{p}_2$, such that $\delta(\overline{r}_1, \overline{r}_2) = (\overline{p}_1, \overline{p}_2)$. If an agent in state $\overline{r}_1$ interacts with an agent in state $\overline{r}_2$, then they change states to $\overline{p}_1$ and $\overline{p}_2$. This paper typically defines a protocol by a list of transitions, with δ implicit. There is a *null* transition $\delta(\overline{r}_1, \overline{r}_2) = (\overline{r}_1, \overline{r}_2)$ if a different output for $\delta(\overline{r}_1, \overline{r}_2)$ is not specified.

Given $\mathbf{c} \in \mathbb{N}^\Lambda$ and transition $\tau : \overline{r}_1, \overline{r}_2 \to \overline{p}_1, \overline{p}_2$, we say that τ is *applicable* to $\mathbf{c}$ if $\mathbf{c} \geq \{\overline{r}_1, \overline{r}_2\}$, i.e., $\mathbf{c}$ contains 2 agents, one in state $\overline{r}_1$ and one in state $\overline{r}_2$. If τ is applicable to $\mathbf{c}$, then write $\tau(\mathbf{c})$ to denote the configuration $\mathbf{c} - \{\overline{r}_1, \overline{r}_2\} + \{\overline{p}_1, \overline{p}_2\}$ (i.e., that results from applying τ to $\mathbf{c}$); otherwise $\tau(\mathbf{c})$ is undefined. A finite or infinite sequence of transitions (τ_i) is a *transition sequence*. Given a $\mathbf{c}_0 \in \mathbb{N}^\Lambda$ and a transition sequence (τ_i), the induced *execution sequence* (or *path*) is a finite or infinite sequence of configurations $(\mathbf{c}_0, \mathbf{c}_1, \ldots)$ such that, for all $i \geq 1$, $\mathbf{c}_i = \tau_{i-1}(\mathbf{c}_{i-1})$. If a finite execution sequence, with associated transition sequence $\overline{q}$, starts with $\mathbf{c}$ and ends with $\mathbf{c}'$, we write $\mathbf{c} \Longrightarrow_q \mathbf{c}'$. We write $\mathbf{c} \Longrightarrow_\mathcal{P} \mathbf{c}'$ (or $\mathbf{c} \Longrightarrow \mathbf{c}'$ when $\mathcal{P}$ is clear from context) if such a path exists (i.e., it is possible to reach from $\mathbf{c}$ to $\mathbf{c}'$) and we say that $\mathbf{c}'$ is *reachable* from $\mathbf{c}$. Let $\mathsf{post}_\mathcal{P}(\mathbf{c}) = \{\mathbf{c}' \mid \mathbf{c} \Longrightarrow_\mathcal{P} \mathbf{c}'\}$ to denote the set of

all configurations reachable from $\mathbf{c}$, writing $\mathsf{post}(\mathbf{c})$ when $\mathcal{P}$ is clear from context. If it is understood from context what is the initial configuration $\mathbf{i}$, then say $\mathbf{c}$ is simply *reachable* if $\mathbf{i} \Longrightarrow \mathbf{c}$. If a transition $\tau : \overline{r}_1, \overline{r}_2 \to \overline{p}_1, \overline{p}_2$ has the property that for $i \in \{1, 2\}$, $\overline{r}_i \notin \{\overline{p}_1, \overline{p}_2\}$, or if ($\overline{r}_1 = \overline{r}_2$ and ($\overline{r}_i \neq \overline{p}_1$ or $\overline{r}_i \neq \overline{p}_2$)), then we say that τ *consumes* $\overline{r}_i$; i.e., applying τ reduces the count of $\overline{r}_i$. We say τ *produces* $\overline{p}_i$ if it increases the count of $\overline{p}_i$.

2.2 Time Complexity

The model used to analyze time complexity is a discrete-time Markov process, whose states correspond to configurations of the population protocol. In any configuration the next interaction is chosen by selecting a pair of agents uniformly at random and applying transition function δ to determine the next configuration. Since a transition may be null, self-loops are allowed. To measure time we count the expected total number of interactions (including null), and divide by the number of agents n. (In the population protocols literature, this is often called "parallel time"; i.e. n interactions among a population of n agents corresponds to one unit of time). Let $\mathbf{c} \in \mathbb{N}^\Lambda$ and $C \subseteq \mathbb{N}^\Lambda$. Denote the probability that the protocol reaches from $\mathbf{c}$ to some configuration $\mathbf{c}' \in C$ by $\mathsf{Pr}[\mathbf{c} \Longrightarrow C]$. If $\mathsf{Pr}[\mathbf{c} \Longrightarrow C] = 1$, define the *expected time to reach from* $\mathbf{c}$ *to* C, denoted $\mathsf{T}[\mathbf{c} \Longrightarrow C]$, to be the expected number of interactions to reach from $\mathbf{c}$ to some $\mathbf{c}' \in C$, divided by the number of agents $n = \|\mathbf{c}\|$. If $\mathsf{Pr}[\mathbf{c} \Longrightarrow C] < 1$ then $\mathsf{T}[\mathbf{c} \Longrightarrow C] = \infty$.

3 Exact computation of nonlinear functions

In Section 4, we obtained a precise characterization of the linear functions stably computable in sublinear time by population protocols and furthermore show that those not exactly computable in sublinear time are not even approximable with sublinear error in sublinear time. However, the class of functions stably computable (in any amount of time) by population protocols is known to contain non-linear functions such as $f(m_1, m_2) = \max(m_1, m_2)$, or $f(m) = m$ if m is even and $f(m) = 2m$ if m is odd. In fact a function is stably computable by a population protocol if and only if its *graph* $\{(\mathbf{m}, f(\mathbf{m})) \mid \mathbf{m} \in \mathbb{N}^k\}$ is a semilinear set [4, 11]. A set $A \subseteq \mathbb{N}^k$ is *semilinear* if and only if [15] it is expressible as a finite number of unions, intersections, and complements of sets of one of the following two forms: *threshold sets* of the form $\{\mathbf{x} \mid \sum_{i=1}^k a_i \cdot \mathbf{x}(i) < b\}$ for some constants $a_1, \ldots, a_k, b \in \mathbb{Z}$ or *mod sets* of the form $\{\mathbf{x} \mid \sum_{i=1}^k a_i \cdot \mathbf{x}(i) \equiv b \mod c\}$ for some constants $a_1, \ldots, a_k, b, c \in \mathbb{N}$. Say that a set $P \subseteq \mathbb{N}^k$ is a *periodic coset* if there exist $\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_l \in \mathbb{N}^k$ such that $P = \{\mathbf{b} + n_1\mathbf{p}_1 + \ldots + n_l\mathbf{p}_l \mid n_1, \ldots, n_l \in \mathbb{N}\}$. (These are typically called "linear" sets, but we wish to avoid confusion with linear functions.) Equivalently, a set is semilinear if and only if it is a finite union of periodic cosets. We say a function $f : \mathbb{N}^k \to \mathbb{N}$ is *semilinear* if its *graph* $\{(\mathbf{m}, f(\mathbf{m})) \mid \mathbf{m} \in \mathbb{N}^k\} \subset \mathbb{N}^{k+1}$ is a semilinear set. A function f is stably computable by a population protocol (given unbounded time) if and only if f is semilinear [11, 4].

Although our technique fails to completely characterize the efficient computability of all semilinear functions, we show that a wide class of semilinear functions cannot be stably computed in sublinear time: functions that are not eventually $\mathbb{N}$-linear. The only exceptions, for which we cannot prove linear time is required, yet neither is there known a counterexample protocol stably computing the function in sublinear time, are functions whose "non-integral-linearities are near the boundary of $\mathbb{N}^k$". For example, the function $f(m) = 0$ if $m \leq 3$ and $f(m) = m$ otherwise is non-linear (although it is semilinear, so stably computable), but restricted to the domain of inputs > 3, it is linear with positive integer coefficients. Thus it is an example of a function whose "population protocol time complexity" is unknown.

Corollary 4.2 and Observation 5.1 imply that a linear function is stably computable in sublinear time by a population protocol if and only if it is $\mathbb{N}$-linear. Theorem 3.1 generalizes the forward direction (restricted to nonlinear functions) to eventually-$\mathbb{N}$-linear functions.

We first give a formal definition of function computation by population protocols. A *function-computing population protocol* is a tuple $\mathcal{C} = (\Lambda, \delta, \Sigma, \overline{y}, \overline{q})$, where (Λ, δ) is a population protocol, $\Sigma = \{\overline{x}_1, \ldots, \overline{x}_k\} \subset \Lambda$ is the set of *input states*, $\overline{y} \in \Lambda$ is the *output state*, and $\overline{q} \in \Lambda \setminus \Sigma$ is the *quiescent state*. We say that a configuration $\mathbf{o} \in \mathbb{N}^\Lambda$ is *stable* if, for all $\mathbf{o}' \in \mathsf{post}(\mathbf{o})$, $\mathbf{o}(\overline{y}) = \mathbf{o}'(\overline{y})$, i.e., the count of $\overline{y}$ cannot change once $\mathbf{o}$ is reached.

Let $f : \mathbb{N}^k \to \mathbb{N}$, $\mathbf{i} \in \mathbb{N}^\Lambda$, and let $\mathbf{m} = \mathbf{i} \restriction \Sigma$. We say that $\mathcal{C}$ *stably computes* f *from* $\mathbf{i}$ if, for all $\mathbf{c} \in \mathsf{post}(\mathbf{i})$, there exists a stable $\mathbf{o} \in \mathsf{post}(\mathbf{c})$ such that $\mathbf{o}(\overline{y}) = f(\mathbf{m})$, i.e., $\mathcal{C}$ stabilizes to the correct output from the initial configuration $\mathbf{i}$. However, for any input $\mathbf{m} \in \mathbb{N}^k$, there are many initial configurations $\mathbf{i} \in \mathbb{N}^\Lambda$ representing it (i.e., such that $\mathbf{i} \restriction \Sigma = \mathbf{m}$). We now formalize what sort of initial configurations $\mathcal{C}$ is required to handle.

We say a function $q_0 : \mathbb{N}^k \to \mathbb{N}$ is *linearly bounded* if there is a constant $c \in \mathbb{N}$ such that, for all $\mathbf{m} \in \mathbb{N}^k$, $q_0(\mathbf{m}) \leq c\|\mathbf{m}\|$. We say that $\mathcal{C}$ *stably computes* f if there is a linearly bounded function $q_0 : \mathbb{N}^k \to \mathbb{N}$ such that, for any $\mathbf{i} \in \mathbb{N}^\Lambda$, defining $\mathbf{m} = \mathbf{i} \restriction \Sigma$, if $\mathbf{i}(\overline{q}) \geq q_0(\mathbf{m})$ and $\mathbf{i}(\overline{s}) = 0$ for all $\overline{s} \in \Lambda \setminus (\Sigma \cup \{\overline{q}\})$, then $\mathcal{C}$ stably computes f from $\mathbf{i}$. It is well-known[6] that this is equivalent to requiring, under the randomized model in which the next interaction is between a pair of agents picked uniformly at random, that the protocol stabilizes on the correct output with probability 1. More formally, given $f : \mathbb{N}^k \to \mathbb{N}$ and $\mathbf{m} \in \mathbb{N}^k$, defining $S^{\mathcal{C}}_{f,\mathbf{m}} = \{\mathbf{o} \in \mathbb{N}^\Lambda \mid \mathbf{o} \text{ is stable and } \mathbf{o}(\overline{y}) = f(\mathbf{m})\}$, $\mathcal{C}$ stably computes f if and only if, for all $\mathbf{m}$, defining $\mathbf{i}$ with $\mathbf{i} \restriction \Sigma = \mathbf{m}$ as above with $\mathbf{i}(\overline{q})$ sufficiently large, $\mathsf{Pr}\left[\mathbf{i} \Longrightarrow S^{\mathcal{C}}_{f,\mathbf{m}}\right] = 1$. It is also equivalent to requiring that every fair infinite execution leads to a correct stable configuration, where an execution is *fair* if every configuration infinitely often reachable appears infinitely often in the execution. We say that an initial configuration $\mathbf{i}$ so defined is *valid*. Since all semilinear functions are linearly bounded [11], a linearly bounded q_0 suffices to ensure there are enough agents to represent the output of a semilinear function, even if we choose $\mathbf{i}(\overline{q}) = q_0(\mathbf{i} \restriction \Sigma)$. If q_0 were *not* linearly bounded, and thus a super-linear count of state $\overline{q}$ is required, we would essentially need to do non-semilinear computation just to initialize the population protocol.

Let $f : \mathbb{N}^k \to \mathbb{N}$ and $t : \mathbb{N} \to \mathbb{N}$. Given a function-computing population protocol $\mathcal{C}$ that stably computes f, we say $\mathcal{C}$ *stably computes* f *in expected time* t if, for all valid initial configurations $\mathbf{i}$ of $\mathcal{C}$, letting $\mathbf{m} = \mathbf{i} \restriction \Sigma$, $\mathsf{T}\left[\mathbf{i} \Longrightarrow S^{\mathcal{C}}_{f,\mathbf{m}}\right] \leq t(n)$.

▶ **Theorem 3.1.** *Let $f : \mathbb{N}^k \to \mathbb{N}$, and let $\mathcal{C}$ be a function-computing population protocol that stably computes f. If f is not eventually-$\mathbb{N}$-linear then $\mathcal{C}$ takes expected time $\Omega(n)$.*

Techniques developed in previous work for proving time lower bounds [14, 1] can certainly generalize beyond leader election and majority, although it was not clear what precise category of computation they cover. However, to extend the impossibility results to all not eventually-$\mathbb{N}$-linear functions, we needed to develop new tools.

Both in prior and current work, the high level intuition of the proof technique is as follows. The overall argument is a proof by contradiction: if sublinear time computation is possible then we find a nefarious execution sequence which stabilizes to an incorrect output. In more detail, sublinear time computation requires avoiding "bottlenecks"—having to go through a transition in which both states are present in small count (constant independent of the number of agents n). Traversing even a single such transition requires linear time. Technical lemmas show that bottleneck-free execution sequences from α-dense initial configurations (i.e., initial configurations where every state that is present is present in at least αn count)

are amenable to predictable "surgery" [14, 1]. At the high level, the surgery lemmas show how states that are present in "low" count when the population protocol stabilizes, can be manipulated (added or removed) such that only "high" count other states are affected. Since it can also be shown that changing high count states in a stable configuration does not affect its stability, this means that the population protocol cannot "notice" the surgery, and remains stabilized to the previous output. For leader election, the surgery allows one to remove an additional leader state (leaving us with no leaders). For majority computation [1], the input in the minority must be present in low count (or absent) at the end. This allows one to add enough of the minority input to turn it into the majority, while the protocol continues to output the wrong answer.

However, applying the previously developed surgery lemmas to fool a more general function computing population protocol is more difficult. The surgery to consume additional input states affects the count of the output state, which could be present in "large count" at the end. How do we know that the effect of the surgery on the output is not consistent with the desired output of the function? In order to arrive at a contradiction we develop two new techniques, both of which are necessary to cover all cases. The first involves showing that the slope of the change in the count of the output state as a function of the input states is inconsistent. The second involves exposing the semilinear structure of the graph of the function being computed, and forcing it to enter the "wrong piece" (i.e., periodic coset).

4 Sublinear-time, sublinear-error approximation of linear functions with negative or non-integer coefficients is impossible

A *function-approximating population protocol* is a tuple $\mathcal{A} = (\Lambda, \delta, \Sigma, \overline{y}, \overline{q}, \overline{a})$, where $(\Lambda, \delta, \Sigma, \overline{y}, \overline{q})$ is a function-computing population protocol and $\overline{a} \in \Lambda \setminus (\Sigma \cup \{\overline{y}, \overline{q}\})$ is the *approximation state*. Let $\epsilon, \tau \in \mathbb{N}$; intuitively τ represents the "target" (or "true") function output, and ϵ represents the allowed approximation error. We say that a configuration $\mathbf{o} \in \mathbb{N}^\Lambda$ is ϵ-τ-*correct* if $|\mathbf{o}(\overline{y}) - \tau| \leq \epsilon$.

Let $f : \mathbb{N}^k \to \mathbb{N}$, $\epsilon \in \mathbb{N}$, $\mathbf{i} \in \mathbb{N}^\Lambda$, and let $\mathbf{m} = \mathbf{i} \upharpoonright \Sigma$. We say that $\mathcal{A}$ *stably* ϵ-*approximates* f *from* $\mathbf{i}$ if, for all $\mathbf{c} \in \mathsf{post}(\mathbf{i})$, there exists a $\mathbf{o} \in \mathsf{post}(\mathbf{c})$ that is stable and ϵ-$f(\mathbf{m})$-correct, i.e., from the initial configuration $\mathbf{i}$, $\mathcal{A}$ gets the output to stabilize to a value at most ϵ from the correct output. Let $S^{\mathcal{A}}_{f,\mathbf{m},\epsilon} = \{\mathbf{o} \in \mathbb{N}^\Lambda \mid \mathbf{o}$ is stable and ϵ-$f(\mathbf{m})$-correct $\}$. Note that $\mathcal{A}$ stably ϵ-approximates f from $\mathbf{i}$ if and only if $\mathsf{Pr}\left[\mathbf{i} \Longrightarrow S^{\mathcal{A}}_{f,\mathbf{m},\epsilon}\right] = 1$.

Let $\mathcal{E} : \mathbb{N} \to \mathbb{N}$; the choice of $\mathcal{E}$ as a function instead of a constant reflects the idea that the approximation error is allowed to depend on the initial count $\mathbf{i}(\overline{a})$ of the approximation state $\overline{a}$, i.e., $\mathcal{E}(\mathbf{i}(\overline{a}))$ is the desired approximation error. We say that $\mathcal{A}$ *stably* $\mathcal{E}$-*approximates* f if there are $a_0 \in \mathbb{N}$ and linearly bounded $q_0 : \mathbb{N}^{k+1} \to \mathbb{N}$ such that, for any $\mathbf{i} \in \mathbb{N}^\Lambda$, defining $\mathbf{m} = \mathbf{i} \upharpoonright \Sigma$, if $\mathbf{i}(\overline{a}) \geq a_0$, $\mathbf{i}(\overline{q}) \geq q_0(\mathbf{m}, \mathbf{i}(\overline{a}))$, and $\mathbf{i}(\overline{s}) = 0$ for all $\overline{s} \in \Lambda \setminus (\Sigma \cup \{\overline{q}, \overline{a}\})$, then $\mathcal{A}$ stably $\mathcal{E}(\mathbf{i}(\overline{a}))$-approximates f from $\mathbf{i}$.[3] An initial configuration $\mathbf{i}$ so defined is *valid*.

As we consider leaderless population protocols, we need to make sure that $\overline{a}$ does not act as a small count "leader". Consistent with the rest of this paper, we reason about initial configurations with $\mathbf{i}(\overline{a}) \geq \alpha n$ for some $\alpha > 0$ to ensure α-density.

Let $f : \mathbb{N}^k \to \mathbb{N}$. In defining running time for function-approximating population protocols, we express the expected time as a function of *both* the total number of agents

[3] I.e., the initial count $\mathbf{i}(\overline{a})$ can influence the initial required count $\mathbf{i}(\overline{q})$, since adding more initial $\overline{a}$ may imply that more quiescent agents are required as "fuel". However, a_0 is constant, not a function of $\mathbf{m}$.

$n = \|\mathbf{i}\|$ *and* the initial count $\mathbf{i}(\overline{a})$ of approximation states. Let $\mathcal{E} : \mathbb{N} \to \mathbb{N}$ and $t : \mathbb{N}^2 \to \mathbb{N}$. Given a function-approximating population protocol $\mathcal{A}$ that $\mathcal{E}$-approximates f, we say $\mathcal{A}$ $\mathcal{E}$-approximates f *in expected time* t if, for all valid initial configurations $\mathbf{i}$ of $\mathcal{A}$, letting $\mathbf{m} = \mathbf{i} \upharpoonright \Sigma$, $\mathsf{T}\left[\mathbf{i} \Longrightarrow S^{\mathcal{A}}_{f,\mathbf{m},\mathcal{E}(\mathbf{i}(\overline{a}))}\right] \leq t(n, \mathbf{i}(\overline{a}))$.

The following theorem states that given any linear function f and any population protocol $\mathcal{P}$, if f has a non-integer or negative coefficient, then $\mathcal{P}$ requires at least linear time to approximate f with sublinear error. It states this by contrapositive: if the protocol takes sublinear time, then the error $\mathcal{E} : \mathbb{N} \to \mathbb{N}$ must grow at least linearly with the initial count of approximation state $\overline{a}$. In particular, the initial configurations $\mathbf{i}$ (letting $n = \|\mathbf{i}\|$) on which our argument maximizes the error have $\mathbf{i}(\overline{a}) = \Omega(n)$. Thus, the fact that $\mathcal{E}(a) \geq \gamma a$ implies that on these $\mathbf{i}$, the error is $\Omega(n)$.

▶ **Theorem 4.1.** *Let $f : \mathbb{N}^k \to \mathbb{N}$ be a linear function that is not $\mathbb{N}$-linear. Let $\mathcal{E} : \mathbb{N} \to \mathbb{N}$. Let $\mathcal{A}$ be a function-approximating population protocol that stably $\mathcal{E}$-approximates f in expected time t, where for some $\alpha > 0$, $t(n, \alpha n) = o(n)$. Then there is a constant $\gamma > 0$ such that, for infinitely many $a \in \mathbb{N}$, $\mathcal{E}(a) \geq \gamma a$.*

A protocol stably computing f also stably $\mathcal{E}$-approximates f for $\mathcal{E}(a) = 0$, so we have:

▶ **Corollary 4.2.** *Let $f : \mathbb{N}^k \to \mathbb{N}$ be a linear function $f(\mathbf{m}) = \sum_{i=1}^{k} \lfloor c_i \mathbf{m}(i) \rfloor$, where $c_i \notin \mathbb{N}$ for some $i \in \{1, \ldots, k\}$. Let $\mathcal{C}$ be a function-computing population protocol that stably computes f. Then $\mathcal{C}$ takes expected time $\Omega(n)$.*

This gives a complete classification of the asymptotic efficiency of computing linear functions $f(\mathbf{m}) = \sum_{i=1}^{k} \lfloor c_i \mathbf{m}(i) \rfloor$ with population protocols. If $c_i \in \mathbb{N}$ for all $i \in \{1, \ldots, k\}$, then f is stably computable in logarithmic time by Observation 5.1. Otherwise, f requires linear time to stably compute by Corollary 4.2.

5 Logarithmic-time, linear-error approximation of linear functions with nonnegative rational coefficients is possible

It is easy to see that any $\mathbb{N}$-linear function f can be stably computed in logarithmic time. Recall that $\overline{x}, \overline{q} \to \overline{y}, \overline{y}$ stably computes $f(m) = 2m$ in expected time $O(\log n)$. The extension to larger coefficients and multiple inputs is routine:

▶ **Observation 5.1.** *Let $f : \mathbb{N}^k \to \mathbb{N}$ be an $\mathbb{N}$-linear function. There is a function-computing population protocol that stably computes f in expected time $O(\log n)$.*

We now describe how to stably approximate linear functions with nonnegative *rational* coefficients, i.e., $\mathbb{Q}_{\geq 0}$-linear functions, with a linear approximation error, in logarithmic time. (It is open to do this for negative coefficients, e.g., $f(m_1, m_2) = m_1 - m_2$). Recall the following simple example of a population protocol that approximately divides by 2 (that is, with probability 1 it outputs a value guaranteed to be a certain distance to the correct output), with a linear approximation error, and is fast ($O(\log n)$ time) with initial counts $\#\overline{x} = m$, $\#\overline{a} = \gamma m$, and $\#\overline{q} = \#\overline{y} = 0$:

$$\overline{a}, \overline{x} \to \overline{b}, \overline{y}$$
$$\overline{b}, \overline{x} \to \overline{a}, \overline{q}$$

which stabilizes $\#\overline{y}$ to somewhere in the interval $\{m/2, m/2 + 1, \ldots, m/2 + \gamma m\}$.

To see that the protocol is correct, note that the transition sequence can make $\#\overline{y}$ closer to one endpoint of the interval or the other depending on which transitions are chosen to consume the *last* γm of $\overline{x}$, but no matter what, the first transition executes at least as many times as the second, but not more than γm times more.

If $\#\overline{a} = 1$ initially, the above protocol stably computes $\lfloor m/2 \rfloor$ (taking linear time just for the last transition; and in total takes $\Theta(n \log n)$ time, by a coupon collector argument).

To see that the protocol takes $O(\log n)$ time if $\#\overline{a} = \gamma m$ initially, note $n = m + \gamma m \leq 2m$. Observe that $\#\overline{a} + \#\overline{b} = \gamma m$ in any reachable configuration. Thus the probability any given interaction is one of the above two transitions is $\approx \frac{\gamma m \#\overline{x}}{n^2}$, so the expected number of interactions until such a transition occurs is $\frac{n^2}{\gamma m \#\overline{x}}$. After m such transitions occur, all the input $\overline{x}$ is gone and the protocol stabilizes, which by linearity of expectation takes expected number of interactions

$$\sum_{\#\overline{x}=1}^{m} \frac{n^2}{\gamma m \#\overline{x}} = \frac{n^2}{\gamma m} \sum_{\#\overline{x}=1}^{m} \frac{1}{\#\overline{x}} \approx \frac{n^2}{\gamma m} \ln m \leq \frac{n^2}{\gamma n/2} \ln n = \frac{2n}{\gamma} \ln n,$$

i.e., expected parallel time $\frac{2}{\gamma} \ln n$. Thus this shows a tradeoff between accuracy and speed in a single protocol, adjustable by the initial count of $\overline{a}$. In this case, the approximation error increases, and the expected time to stabilization decreases, with increasing initial $\#\overline{a}$.

More generally, we can prove the following. In particular, if $a = \Omega(n)$, then $t(n, a) = O(\log n)$. Also, if $a = o(n)$, then the approximation error is $o(n)$, and if $a = \omega(\log n)$, then the expected time is $o(n)$ also. This does not contradict Theorem 4.1 since setting $a = o(n)$ implies the initial configurations are not all α-dense for a fixed $\alpha > 0$.

▶ **Theorem 5.2.** *Let $f : \mathbb{N}^k \to \mathbb{N}$ be a $\mathbb{Q}_{\geq 0}$-linear function. Let $\mathcal{E} : \mathbb{N} \to \mathbb{N}$ be the identity function. Define $t : \mathbb{N}^2 \to \mathbb{N}$ by $t(n, a) = \frac{n}{a} \log n$. Then there is a function-approximating population protocol $\mathcal{A}$ that $\mathcal{E}$-approximates f in expected time $O(t)$.*

The basic analysis is similar to the example protocol above, and the extension to rational coefficients other than $\frac{1}{2}$ follows techniques used in similar papers on function computation with chemical reaction networks [11, 13].

6 Predicate computation

In this section we show that a wide class of Boolean predicates cannot be stably computed in sublinear time by population protocols (without a leader). Intuitively, this is the class of predicates $\phi : \mathbb{N}^k \to \{0, 1\}$ such that for all $m \in \mathbb{N}$, there are two inputs $\mathbf{m}_0, \mathbf{m}_1 \in \mathbb{N}^k_{\geq m}$ such that $\phi(\mathbf{m}_0) \neq \phi(\mathbf{m}_1)$. (See the definition of *eventually constant* below.)

Formally, a *predicate-deciding population protocol* is a tuple $\mathcal{D} = (\Lambda, \delta, \Sigma, \Upsilon_1)$, where (Λ, δ) is a population protocol, $\Sigma \subseteq \Lambda$ is the set of *input states*, and $\Upsilon_1 \subseteq \Lambda$ is the set of *1-voters*. By convention, we define $\Upsilon_0 = \Lambda \setminus \Upsilon_1$ to be the set of *0-voters*. The *output* $\Phi(\mathbf{c})$ of a configuration $\mathbf{c} \in \mathbb{N}^\Lambda$ is $b \in \{0, 1\}$ if $\mathbf{c}(\overline{s}) = 0$ for all $\overline{s} \in \Upsilon_{1-b}$ (i.e., if the vote is unanimously b); the output is undefined if voters of both types are present. We say $\mathbf{o} \in \mathbb{N}^\Lambda$ is *stable* if $\Phi(\mathbf{o})$ is defined and for all $\mathbf{o}' \in \mathsf{post}(\mathbf{o})$, $\Phi(\mathbf{o}') = \Phi(\mathbf{o})$. For all $\mathbf{m} \in \mathbb{N}^k$, define initial configuration $\mathbf{i}_\mathbf{m} \in \mathbb{N}^\Lambda$ by $\mathbf{i}_\mathbf{m} \upharpoonright \Sigma = \mathbf{m}$ and $\mathbf{i}_\mathbf{m} \upharpoonright (\Lambda \setminus \Sigma) = \mathbf{0}$. Call such an initial configuration *valid*. For any valid initial configuration $\mathbf{i}_\mathbf{m} \in \mathbb{N}^\Lambda$ and predicate $\phi : \mathbb{N}^k \to \{0, 1\}$, let $S_{\mathbf{i}_\mathbf{m}, \phi} = \{\mathbf{o} \in \mathbb{N}^\Lambda \mid \mathbf{i}_\mathbf{m} \Longrightarrow \mathbf{o}, \mathbf{o} \text{ is stable, and } \Phi(\mathbf{o}) = \phi(\mathbf{m})\}$. A population protocol *stably decides* a predicate $\phi : \mathbb{N}^k \to \{0, 1\}$ if, for any valid initial configuration $\mathbf{i}_\mathbf{m} \in \mathbb{N}^\Lambda$, $\mathsf{Pr}[\mathbf{i}_\mathbf{m} \Longrightarrow S_{\mathbf{i}_\mathbf{m}, \phi}] = 1$. This is equivalent to requiring that for all $\mathbf{c} \in \mathsf{post}(\mathbf{i}_\mathbf{m})$, there is $\mathbf{o} \in \mathsf{post}(\mathbf{c})$ such that $\mathbf{o}$ is stable and $\Phi(\mathbf{o}) = \phi(\mathbf{m})$.

For example, the protocol defined by transitions

$$\overline{x}_1, \overline{x}_2 \to \overline{q}_1, \overline{q}_2$$
$$\overline{x}_1, \overline{q}_2 \to \overline{x}_1, \overline{q}_1$$
$$\overline{x}_2, \overline{q}_1 \to \overline{x}_2, \overline{q}_2$$
$$\overline{q}_1, \overline{q}_2 \to \overline{q}_1, \overline{q}_1$$

if $\Upsilon_1 = \{\overline{x}_1, \overline{q}_1\}$ and $\Upsilon_0 = \{\overline{x}_2, \overline{q}_2\}$, decides whether $m_1 = \mathbf{i}(\overline{x}_1) \geq m_2 = \mathbf{i}(\overline{x}_2)$. The first transition stops once the less numerous input state is gone. If $\overline{x}_1$ (resp. $\overline{x}_2$) is left over, then the second (resp. third) transition converts $\overline{q}_i$ states to its vote. If neither is left over (i.e., if $m_1 = m_2$, requiring output 1), the fourth transition converts all $\overline{q}_2$ states to $\overline{q}_1$.

Let $\phi : \mathbb{N}^k \to \{0,1\}$, and for $b \in \{0,1\}$, define $\phi^{-1}(b) = \{\mathbf{m} \in \mathbb{N}^k \mid \phi(\mathbf{m}) = b\}$ to be the set of inputs on which ϕ outputs b. We say ϕ is *eventually constant* if there is $m_0 \in \mathbb{N}$ such that ϕ is constant on $\mathbb{N}^k_{\geq m_0} = \{\mathbf{m} \in \mathbb{N}^k \mid (\forall i \in \{1, \ldots, k\})\ \mathbf{m}(i) \geq m_0\}$, i.e., either $\phi^{-1}(0) \cap \mathbb{N}^k_{\geq m_0} = \emptyset$ or $\phi^{-1}(1) \cap \mathbb{N}^k_{\geq m_0} = \emptyset$. In other words, although ϕ may have an infinite number of each output, "sufficiently far from the boundary" (where all coordinates exceed m_0), only one output appears.

The following theorem shows that any predicate that is not eventually constant cannot be stably decided in sublinear time by a population protocol.

▶ **Theorem 6.1.** *Let $\phi : \mathbb{N}^k \to \{0,1\}$ and $\mathcal{D}$ be a predicate-deciding population protocol that stably decides ϕ. If ϕ is not eventually constant, then $\mathcal{D}$ takes expected time $\Omega(n)$.*

Alistarh, Aspnes, Eisenstat, Gelashvili, and Rivest [1] showed a linear-time lower bound on any leaderless population protocol deciding the majority predicate. Recall that their technique is based on showing that after adding enough of the input in the minority to change it to the majority, the effect of this addition can be effectively nullified by surgery of the transition sequence, yielding a stable configuration with the original (now incorrect) answer. The technique can be extended easily to show various other specific predicates, such as equality and parity, also require linear time. We use the same technique of finding pairs of inputs with opposite correct answers and apply a similar transition sequence surgery. The main difficulty in showing Theorem 6.1, which covers the class of *all* predicates that are semilinear but not eventually constant, is to identify a common characteristic that can be exploited to find pairs of inputs that are α-dense for some $\alpha > 0$. Here, we rely on the semilinear structure of the predicate computed. Indeed, note that we cannot find such α-dense pairs for the predicate $\phi : \mathbb{N}^2 \to \{0,1\}$ with support $\{(k, 2^k) \mid k \in \mathbb{N}\}$, which is not eventually constant (but also not semilinear).

7 Open Questions

Time complexity of other functions. What is the optimal time complexity of computing semilinear functions and predicates not satisfying the hypotheses of Theorems 3.1 and 6.1; namely the *eventually-$\mathbb{N}$-linear* functions, (e.g., $f(m) = 0$ if $m < 3$ and $f(m) = m$ otherwise) and *eventually-constant* predicates (e.g., $\phi(m) = 1$ iff $m \geq 2$)?

Stabilization vs convergence. Measuring time to stabilization in the randomized model, as we do here, measures the expected time until the probability of changing the output becomes 0. Our proof shows only that stabilization must take expected $\Omega(n)$ time. However, convergence could occur much earlier in a transition sequence than stabilization (we can say

a particular transition sequence *converged* at the point when the output count is the same in every subsequently reached configuration). We conjecture that similar negative results hold for convergence for leaderless population protocols. It is also open whether stabilization can occur in sublinear time, *even with an initial leader*. The known stably computing protocols converging in $O(\log^5 n)$ time [5, 11] provably require expected time $\Omega(n)$ to stabilize.

Acknowledgements. We are grateful to Sungjin Im for the proof of an important technical lemma, and we thank anonymous reviewers for very helpful comments.

References

1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in molecular computation. In *SODA 2017: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017. to appear.

2 Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *ICALP 2015: Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming, Kyoto, Japan*, 2015.

3 Dana Angluin, James Aspnes, Zoë Diamadi, Michael Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18:235–253, 2006. Preliminary version appeared in PODC 2004. doi:10.1007/s00446-005-0138-3.

4 Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *PODC 2006: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 292–299, New York, NY, USA, 2006. ACM Press. doi:10.1145/1146381.1146425.

5 Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008. Preliminary version appeared in DISC 2006.

6 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

7 Alexandre Baccouche, Kevin Montagne, Adrien Padirac, Teruo Fujii, and Yannick Rondelez. Dynamic dna-toolbox reaction circuits: a walkthrough. *Methods*, 67(2):234–249, 2014.

8 James M Bower and Hamid Bolouri. *Computational modeling of genetic and biochemical networks*. MIT press, 2004.

9 E. Cardoza, Richard J. Lipton, and Albert R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups (preliminary report). In *STOC 1976: Proceedings of the 8th annual ACM Symposium on Theory of Computing*, pages 50–54. ACM, 1976.

10 Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. Speed faults in computation by chemical reaction networks. *Distributed Computing*, 2015. to appear. Special issue of invited papers from DISC 2014.

11 Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. Preliminary version appeared in DNA 2012.

12 Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8(10):755–762, 2013.

13 David Doty and Monir Hajiaghayi. Leaderless deterministic chemical reaction networks. *Natural Computing*, 14(2):213–223, 2015. Preliminary version appeared in DNA 2013.

14 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 2016. to appear. Special issue of invited papers from DISC 2015.

15 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966. URL: `http://projecteuclid.org/euclid.pjm/1102994974`.

16 Richard M Karp and Raymond E Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.

17 Ernst W Mayr and Albert R Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics*, 46(3):305–329, 1982.

18 Othon Michail and Paul G Spirakis. How many cooks spoil the soup? In *International Colloquium on Structural Information and Communication Complexity*, pages 3–18. Springer, 2016.

19 Carl A Petri. Communication with automata. Technical report, DTIC Document, 1966.

20 Niranjan Srinivas. *Programming chemical kinetics: Engineering dynamic reaction networks with DNA strand displacement*. PhD thesis, California Institute of Technology, 2015.

21 Chris Thachuk, Erik Winfree, and David Soloveichik. Leakless DNA strand displacement systems. In *DNA 2015: Proceedings of the 21st International Conference on DNA Computing and Molecular Programming*, pages 133–153. Springer, 2015.

22 Vito Volterra. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. *Mem. Acad. Lincei Roma*, 2:31–113, 1926.

23 David Yu Zhang and Georg Seelig. Dynamic dna nanotechnology using strand-displacement reactions. *Nature chemistry*, 3(2):103–113, 2011.